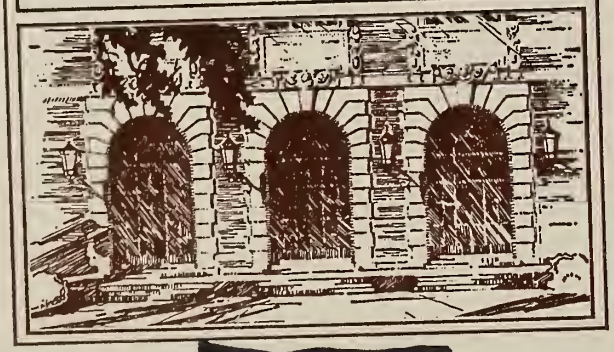


ITERATIVE AND DIRECT METHODS FOR SOLVING POISSON'S EQUATION AND THEIR ADAPTABILITY TO ILLIAC IV

by

James H. Ericksen

December 20, 1972

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-72-574

ITERATIVE AND DIRECT METHODS FOR SOLVING POISSON'S EQUATION AND THEIR ADAPTABILITY TO ILLIAC IV

by

James H. Ericksen

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

December 20, 1972

This work was supported in part by the Advanced Research Project Agency of the Department of Defense and was monitored by the U. S. Army Research Office-Durham, under Contract No. DAH C04-72-C-0001, and supported in part by the Atmospheric Sciences Section, National Science Foundation, NSF Grant GA-31407. Portions of this work were submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, June, 1972.

ACKNOWLEDGEMENT

The author wishes to express his gratitude to Professor Michael S. Sher and Mrs. Masako Ogura for suggestions and comments throughout this project. The author is grateful to Dr. Walter L. Heimerdinger who supplied information dealing with timing on ILLIAC IV. The author wishes to thank Mrs. Nancy Freece for typing this manuscript. Finally, the author is indebted to the Center for Advanced Computation for providing the opportunity to carry out this study.

ABSTRACT

This paper examines iterative and direct methods for solving Poisson's equation with regard to their adaptation to ILLIAC IV. SOR, SLOR, ADI, and FACR are programmed in GLYPNIR. Detailed suggestions on ASK code for these methods are also supplied.

FACR, Fourier Analysis and Cyclic Reduction, is the fastest method on rectangular meshes. SOR, Successive Over Relaxation, seems to be the most promising for nonrectangular meshes. The methods are between thirty and forty-five times faster on ILLIAC IV than on a serial machine with speed equal to one of the ILLIAC IV PEs (Processing Elements).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. POISSON'S EQUATION IN MATRIX FORM

We will consider Poisson's equation with Dirichlet boundary conditions defined on a rectangular region. This equation can be written as

$$\psi(X,Y) = \frac{\partial^2 U(X,Y)}{\partial Y^2} + \frac{\partial^2 U(X,Y)}{\partial X^2} \qquad (1)$$

where the value of U is given at the boundaries $X = 0$, $Y = 0$, $X = (m-1)h_x$ and $Y = (n-1)h_y$. We denote $U((i-1)h_x, (j-1)h_y)$ by $U_{i,j}$, $\psi((i-1)h_x, (j-1)h_y)$ by $\psi_{i,j}$ and error of order p by $O(p)$. By dividing the region into mesh points such that the distance between mesh points in the vertical direction is h_y and in the horizontal direction is h_x we get Figure 1.a. The region in Figure 1.a can be rotated 90^o clockwise to obtain the mesh U in Figure 1.b where $U_{i,j}$ is the mesh point in the $i^{\underline{th}}$ row and the $j^{\underline{th}}$ column.

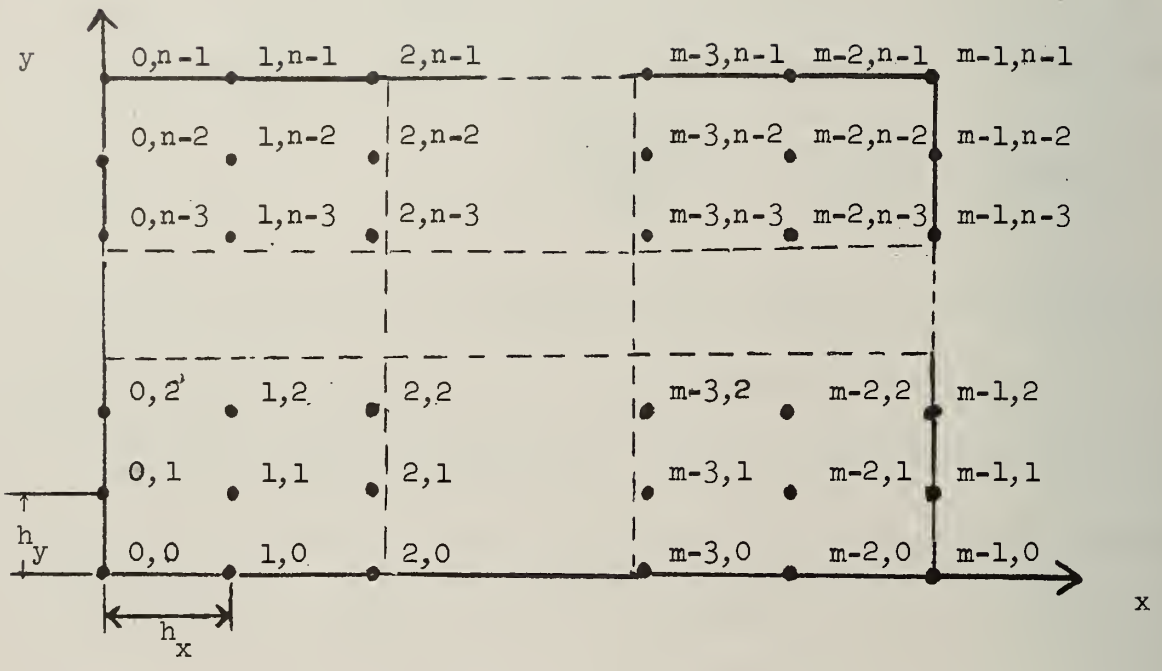
By use of Taylor's Theorem we obtain:

$$U_{i+1,j} = U_{i,j} + h_x \frac{\partial U}{\partial X}(i,j) + \frac{h_x^2}{2}\frac{\partial^2 U}{\partial X^2}(i,j) + \frac{h_x^3}{3!}\frac{\partial^3 U}{\partial X^3}(i,j) + \frac{h_x^4}{4!}\frac{\partial^4 U}{\partial X^4}(i,j) + \cdots$$

$$U_{i-1,j} = U_{i,j} - h_x \frac{\partial U}{\partial X}(i,j) + \frac{h_x^2}{2}\frac{\partial^2 U}{\partial X^2}(i,j) - \frac{h_x^3}{3!}\frac{\partial^3 U}{\partial X^3}(i,j) + \frac{h_x^4}{4!}\frac{\partial^4 U}{\partial X^4}(i,j) + \cdots$$

$$U_{i,j+1} = U_{i,j} + h_y \frac{\partial U}{\partial Y}(i,j) + \frac{h_y^2}{2}\frac{\partial^2 U}{\partial Y^2}(i,j) + \frac{h_y^3}{3!}\frac{\partial^3 U}{\partial Y^3}(i,j) + \frac{h_y^4}{4!}\frac{\partial^4 U}{\partial Y^4}(i,j) + \cdots$$

$$U_{i,j-1} = U_{i,j} - h_y \frac{\partial U}{\partial Y}(i,j) + \frac{h_y^2}{2}\frac{\partial^2 U}{\partial Y^2}(i,j) - \frac{h_y^3}{3!}\frac{\partial^3 U}{\partial Y^3}(i,j) + \frac{h_y^4}{4!}\frac{\partial^4 U}{\partial Y^4}(i,j) + \cdots$$



a) The rectangular region with the mesh points.

$$
\begin{array}{ccccccc}
U_{1,1'} & U_{1,2'} & U_{1,3'} & \cdots & U_{1,n-1'} & & U_{1,n} \\
U_{2,1'} & U_{2,2'} & U_{2,3'} & \cdots & U_{2,n-1'} & & U_{2,n} \\
U_{3,1'} & U_{3,2'} & U_{3,3'} & \cdots & U_{3,n-1'} & & U_{3,n} \\
\vdots & \vdots & \vdots & \cdots & \vdots & & \vdots \\
U_{m-1,1'} & U_{m-1,2} & U_{m-1,3'} & \cdots & U_{m-1,n-1'} & & U_{m-1,n} \\
U_{m,1'} & U_{m,2'} & U_{m,3'} & \cdots & U_{m,n-1'} & & U_{m,n}
\end{array}
$$

b) The mesh points with separation of boundary and interior mesh points.

Figure 1. The Mesh

Thus $\frac{\delta^2 U}{\delta Y^2}(i,j) = \frac{1}{h_y^2}(U_{i,j+1} + U_{i,j-1} - 2U_{i,j}) + O(h_y^2)$ (2)

and $\frac{\delta^2 U}{\delta X^2}(i,j) = \frac{1}{h_x^2}(U_{i+1,j} + U_{i-1,j} - 2U_{i,j}) + O(h_x^2)$ (3)

Substituting (2) and (3) into (1) we obtain

$$\frac{1}{h_y^2}(U_{i,j+1} + U_{i,j-1} - 2U_{i,j}) + \frac{1}{h_x^2}(U_{i+1,j} + U_{i-1,j} - 2U_{i,j}) \approx \psi_{i,j}. \quad (4)$$

This is the equation for any interior mesh point $U_{i,j}$. The values of the mesh points on the boundary are given, and require no equations. Let us take all the equations for the interior mesh points and place the constant terms on the right hand side of the equations. For example, the equation for $U_{2,j}$ where $3 \leq j \leq n-2$ will be

$$\frac{1}{h_y^2}(U_{2,j+1} + U_{2,j-1} - 2U_{2,j}) + \frac{1}{h_x^2}(U_{3,j} - 2U_{2,j}) \approx \psi_{2,j} - \frac{1}{h_x^2} U_{1,j}$$

The equation for $U_{2,2}$ will be

$$\frac{1}{h_y^2}(U_{2,3} - 2U_{2,2}) + \frac{1}{h_x^2}(U_{3,2} - 2U_{2,2}) \approx \psi_{2,2} - \frac{1}{h_x^2} U_{1,2} - \frac{1}{h_y^2} U_{2,1}$$

Combining the constant terms into one term, $M_{i,j}$, for each $U_{i,j}$ we obtain

$$M_{i,j} = \psi_{i,j} - \frac{1}{h_y^2}(U_{i,j+1}\ \delta_{j+1,n} + U_{i,j-1}\ \delta_{j-1,1}) - \frac{1}{h_x^2}(U_{i+1,j}\ \delta_{i+1,m} + U_{i-1,j}\ \delta_{i-1,1}) \quad (5)$$

where $\delta_{k,\ell} = \begin{cases} 0 \text{ if } k \neq \ell \\ 1 \text{ if } k = \ell \end{cases}$.

Consider the internal mesh points as a vector U_I in which the order of the mesh points is as follows:

1) The first interior point in the first row becomes the first element of the vector.

2) All interior points of row i are placed in the vector in the same order they appear in row i.

3) The $(n-1)^{\underline{th}}$ element of the $i^{\underline{th}}$ row is followed by the first interior point of the following row, where $2 \leq i \leq m-2$.

The transpose of U_I equals

$$(U_{2,2}, U_{2,3}, \cdots, U_{2,n-1}; U_{3,2}, U_{3,3}, \cdots, U_{3,n-1}; \cdots ; U_{m-1,2}, U_{m-1,3}, \cdots , U_{m-1,n-1}) .$$

Using this order we assemble the interior mesh point equations and consider them as the matrix equation $AU_I = M$ where M is the vector of constant terms and A is a matrix made up of the coefficients of the mesh points.

The matrix A is a $(m-2)(n-2)$ by $(m-2)(n-2)$ block tridiagonal matrix of the following form:

$$A = \begin{bmatrix} B & C & & & & & \\ C & B & C & & & & \\ & C & B & C & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & C & B & C & \\ & & & & C & B & \end{bmatrix} \qquad (6)$$

where $C = aI_{n-2}$, $\quad a = \dfrac{+1}{h_x^2}$

and B is a (n-2) by (n-2) tridiagonal matrix of the form

$$B = \begin{bmatrix} e & d & & & \\ d & e & d & & \\ & \ddots & \ddots & \ddots & \\ & & d & e & d \\ & & & d & e \end{bmatrix} \qquad (6.1)$$

where $d = \frac{1}{h_y^2}$ and $e = -(2a + 2d)$.

There are two general approaches used in solving this system of linear equations, direct methods and iterative methods.* In this study we have limited our discussion to three popular iterative methods and one direct method: the successive over-relaxation method (SOR), the successive line over-relaxation method (SLOR), the alternating direction implicit method (ADI), and Hockney's direct method (FACR). In the discussions of FACR three types of boundary conditions will be examined: periodic, Neumann and Dirichlet. The discussion of iterative methods considers only Dirichlet boundary conditions.

---

* No matter which method is used in solving the equations the lower bound on the error is $0(\max(h_y^2, h_x^2))$. This is due to the error terms in (2) and (3).

2. IMPLEMENTATION OF ITERATIVE METHODS ON ILLIAC IV

2.1 Introduction to SLOR and SOR

Iterative methods have one thing in common. They each begin with an initial guess to the solution and improve* upon the guess with each iteration. Thus the following notation is useful: The value of U_I after the ℓ^{th} iteration is denoted by $U_I^{(\ell)}$. $U_I^{(0)}$ is the initial guess and if the method converges then $\lim_{\ell \to \infty} (U_I^{(\ell)} - U_I^{(\ell+1)}) = 0$.

Matrix A can be divided into three matrices such that $A = D - E - F$ where D is block diagonal, E is strictly lower triangular and F is strictly upper triangular. Thus $AU_I = M$ can be written as $DU_I - (E + F) U_I = M$. From this we get the iterative method:

$$U_I^{(\ell+1)} = D^{-1} (E + F) U_I^{(\ell)} + D^{-1} M \tag{7}$$

The matrix equation can also be grouped to obtain

$$U_I^{(\ell+1)} = (D-E)^{-1} FU_I^{(\ell)} + (D-E)^{-1} M \tag{8}$$

The asymptotic convergence rate of (8) is twice as fast as (7) [Todd, Page 391]. This is due to the fact that the matrix has Young's property A [Wachspress, Page 102]. (8) is a successive method while (7) is a simultaneous method.** By defining $J = \frac{1}{\omega} (D - \omega E)$ and $H = \frac{1}{\omega} (\omega F + (1-\omega)D)$ for $\omega \neq 0$ we get $\omega JU_I = \omega HU_I + \omega M$ from $AU_I = M = (J-H)U_I$.

---

* Improvement is measured by a reduction in the error vector.

** Successive methods use new information as soon as it is available while simultaneous methods use old values for the entire iteration.

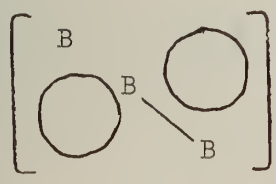
We can write $\omega JU_I = \omega HU_I + \omega M$ as

$$(D-\omega E)U_I^{(\ell+1)} = (\omega F + (1-\omega)D)\, U_I^{(\ell)} + \omega M \qquad (9)$$

Note that (8) and (9) are equivalent if $\omega = 1$. When $0 < \omega < 2$ and A is positive definite, (9) converges. If $1 < \omega < 2$, then (9) is referred to as an over-relaxation method [Forsythe and Wasow, Page 261]. With different values of ω, the convergence rate of (9) is altered. For a single relaxation parameter, it can be shown that $\omega_b = \dfrac{2}{1+\sqrt{1-\rho^2(\beta)}}$, where $\beta = D^{-1}(E+F)$, the Jacobi matrix, and $\rho(\beta)$ is the spectral radius of β, gives the fastest rate of convergence [Young, Page 169].

There are two common ways in which equations (7 - 9) are implemented. The first gives point iterative methods by letting D equal the diagonal of A. Thus (7) becomes Jacobi's method, (8) becomes the formula for successive point relaxation iterative method (SR) and (9) becomes the successive point over relaxation iterative method (SOR).

If we let

$D =$ 

where B is defined in (6.1) then we will get line iterative methods. (7) becomes the simultaneous line iterative method, (8) becomes the

successive line relaxation iterative method (SLR) and (9) becomes the successive line over-relaxation method (SLOR) [Varga, Page 199].

To give the reader a better understanding of the way these methods work, we write the individual equations for the interior mesh points $U_{i,j}$, where $3 \leq i \leq m-2$ and $3 \leq j \leq n-2$ for each method.* First we define h, v, and g as follows:

$$h = \frac{h_y^2}{2(h_x^2 + h_y^2)}, \qquad v = \frac{h_x^2}{2(h_x^2 + h_y^2)}, \qquad \text{and} \quad g = \frac{h_x^2 \, h_y^2}{2(h_x^2 + h_y^2)}.$$

Jacobi's method (simultaneous point iterative method)

$$U_{i,j}^{(\ell+1)} = v\,(U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell)}) + h\,(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell)}) - g\,M_{i,j}. \tag{10}$$

Simultaneous line iterative method

$$U_{i,j}^{(\ell+1)} = v\,(U_{i,j+1}^{(\ell+1)} + U_{i,j-1}^{(\ell+1)}) + h\,(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell)}) - g\,M_{i,j}.$$

Successive point relaxation method (SR)

$$U_{i,j}^{(\ell+1)} = v\,(U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell+1)}) + h\,(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell+1)}) - g\,M_{i,j}.$$

* Note we have excluded the first row, last row, first column, and the last column of the interior mesh points to avoid boundary conditions. These points will be discussed in another section.

Successive line relaxation method (SLR)

$$U_{i,j}^{(\ell+1)} = v\,(U_{i,j+1}^{(\ell+1)} + U_{i,j-1}^{(\ell+1)}) + h\,(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell+1)}) - g\,M_{i,j}\,. \qquad (11)$$

Successive point over-relaxation (SOR)

$$U_{i,j}^{(\ell+1)} = \omega\,v\,(U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell+1)}) + \omega\,h\,(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell+1)}) - \omega\,g\,M_{i,j} + (1-\omega)\,U_{i,j}^{(\ell)}. \qquad (12)$$

Successive line over-relaxation (SLOR)

$$U_{i,j}^{(\ell+1)} = v\,(U_{i,j+1}^{(\ell+1)} + U_{i,j-1}^{(\ell+1)}) + \omega\,h\,(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell+1)}) - \omega\,g\,M_{i,j} + (1-\omega)\,(U_{i,j}^{(\ell)} - v\,(U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell)})). \qquad (13)$$

2.2 Introduction to ADI

Let us look at the equation we are solving once again,

$$\frac{\partial^2 U(X,Y)}{\partial X^2} + \frac{\partial^2 U(X,Y)}{\partial Y^2} = \Psi\,(X,Y) \qquad (1)$$

as before we use Taylor's Theorem and we get

$$\frac{\delta^2 U}{\delta Y^2}(i,j) = \frac{1}{h_y^2}\,(U_{i,j-1} + U_{i,j+1} - 2U_{i,j}) + O(h_y^2) \qquad (2)$$

and

$$\frac{\delta^2 U}{\delta X^2}(i,j) = \frac{1}{h_x^2}\,(U_{i-1,j} + U_{i+1,j} - 2U_{i,j}) + O(h_x^2) \qquad (3)$$

Thus when we form the matrix equation $AU_I = M$ where A is the matrix (6), we are considering a horizontal part of the equation and a vertical part of the equation. We could divide A into two matrices H and V such that $H + V = A$ where V is the matrix containing the coefficients of the vertical equations (2). Both H and V can be placed in tridiagonal form with the appropriate permutation matrix.

We can take the equation

$AU_I = HU_I + VU_I = M$ and obtain

$(H - \omega_{\ell H} I)U_I = M - (V + \omega_{\ell H} I)U_I$ and

$(V - \omega_{\ell V} I)U_I = M - (H + \omega_{\ell V} I)U_I$.

Thus we can get the Peaceman-Rachford ADI scheme [Varga, Page 212]. *

$$U_I^{(\ell+1/2)} = (H - \omega_{\ell H} I)^{-1} [M - (V + \omega_{\ell H} I)U_I^{(\ell)}] \tag{14.1}$$

$$U_I^{(\ell+1)} = (V - \omega_{\ell V} I)^{-1} [M - (H + \omega_{\ell V} I)U_I^{(\ell+1/2)}] \tag{14.2}$$

The parameters, $\omega_{\ell H}$ and $\omega_{\ell V}$ are defined in [Wachspress, Page 192].

It will help to have the corresponding equations for the individual mesh points, $U_{i,j}$, where $3 \le i \le n-2$ and $3 \le j \le m-2$.

$$U_{i,j}^{(\ell+1/2)} = \frac{-1}{\omega_{\ell H} + \frac{2}{h_x^2}} \left[M_{i,j} - \frac{1}{h_y^2} \left(U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell)} \right) - \left(\omega_{\ell H} - \frac{2}{h_y^2} \right) U_{i,j}^{(\ell)} \right.$$

$$\left. - \frac{1}{h_x^2} \left(U_{i-1,j}^{(\ell+1/2)} + U_{i+1,j}^{(\ell+1/2)} \right) \right]. \tag{15.1}$$

---

* ADI requires that A be positive definite. This can be obtained by using $-AU_I = -M$ or by using $-\omega_{\ell V}$ and $-\omega_{\ell H}$ instead of $\omega_{\ell V}$ and $\omega_{\ell H}$. We use the latter technique.

$$U_{i,j}^{(\ell+1)} = \frac{-1}{(\omega_{\ell V}+\frac{2}{h_y^2})} \left[M_{i,j} - \frac{1}{h_x^2}\left(U_{i+1,j}^{(\ell+1/2)} + U_{i-1,j}^{(\ell+1/2)}\right) - \left(\omega_{\ell V} - \frac{2}{h_x^2}\right) U_{i,j}^{(\ell+1/2)}\right.$$

$$\left. - \frac{1}{h_y^2}\left(U_{i,j-1}^{(\ell+1)} + U_{i,j+1}^{(\ell+1)}\right)\right]. \qquad (15.2)$$

Examining (15.1) and (15.2) we note a strong resemblance to simultaneous column relaxation followed by simultaneous row relaxation.

2.3 Implementation of SOR on ILLIAC IV

2.3.1 A Parallel Processor's Effect on the Algorithm

A parallel machine allows operations to be performed simultaneously which are done in separate time intervals on a conventional machine. This is referred to as overlap. The amount of overlap is dependent on the way in which the algorithm is programmed. The SOR algorithm supplies an illustration of how the flow of an algorithm is changed when programmed to maximize overlap on a parallel machine.

Let us examine the equation for SOR closer:

$$U_{i,j}^{(\ell+1)} = \omega\ v\ (U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell+1)}) + \omega\ h\ (U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell+1)}) - \omega\ g\ M_{i,j} \qquad (12)$$
$$+ (1-\omega)\ U_{i,j}^{(\ell)}$$

for $3 \leq i \leq m-2$ and $3 \leq j \leq n-2$. Note the interior points which are adjacent to one or more boundary points are not included. Also note that for the above mesh points $M_{i,j} = \Psi_{i,j}$. The equation for $U_{2,2}$ would be

$$U_{2,2}^{(\ell+1)} = \omega\ v\ U_{2,3}^{(\ell)} + \omega\ h\ U_{3,2}^{(\ell)} - \omega\ g\ M_{2,2} + (1-\omega)\ U_{2,2}^{(\ell)} \qquad (16)$$

where

$$M_{2,2} = \Psi_{2,2} - \frac{1}{h_x^2}\ U_{1,2} - \frac{1}{h_y^2}\ U_{2,1}\ .$$

Thus

$$g\ M_{2,2} = g\ \Psi_{2,2} - v\ U_{2,1} - h\ U_{1,2}\ .$$

This enables us to rewrite (16) as

$$U_{2,2}^{(\ell+1)} = \omega\, v\, (U_{2,3}^{(\ell)} + U_{2,1}^{(\ell+1)}) + \omega\, h\, (U_{3,2}^{(\ell)} + U_{1,2}^{(\ell+1)}) - \omega\, g\, \Psi_{2,2} \qquad (17)$$

To calculate $U_{2,2}$ using (17) takes two more additions than using (16) but $M_{2,2}$ is needed to use (16). On a serial machine, one would use equation (16) to improve $U_{2,2}$. The choice is not so straightforward on a parallel machine if when $U_{2,2}$ is calculated in one processing element (PE), $U_{i,j}$ is calculated in another PE where $3 \leq i \leq m-2$ and (or) $3 \leq j \leq n-2$. In this case to calculate $U_{2,2}$ using (16) we would have to turn the appropriate PE off for two additions. By using (17), we would not have to turn off PEs for those additions and would also save the calculation of $M_{2,2}$. A similar argument can be given for all points adjacent to boundary points. Thus the choice of the equation to evaluate a point adjacent to the boundary in one PE depends on what is being done in the other PEs.

2.3.2 Parallelisms in SOR

Examining equation 12, we note that the values of $U_{i-1,j}^{(\ell+1)}$ and $U_{i,j-1}^{(\ell+1)}$ are needed to evaluate $U_{i,j}^{(\ell+1)}$. Now consider the following mesh:

0	0	0	0	0	0
0	1	2	3	4	0
0	2	3	4	5	0
0	3	4	5	6	0
0	4	5	6	7	0
0	0	0	0	0	0

* Whenever $U_{i,j}$ is a boundary point $U_{i,j}^{(\ell)} = U_{i,j}^{(\ell+1)}$ for all ℓ.

0 indicates a boundary point. The interior mesh points are divided into seven sets (1,2,3..., 7) where set i contains all points labeled by i. To improve any point in set i, we must first improve all the points in the union of all sets j such that $j < i$.

In a computer with 4 PEs, one iteration could be completed in 7 time steps where a time step is the amount of time to improve a mesh point in one PE by improving one set in each time step. But the computer could improve 28 points in 7 time steps. To improve the efficiency of the method, we look at what could be done in each time step assuming we had done everything possible up to that time step.

Time Step	1	2	3	4	5	6	7	8	9	10	11
What	1^1	2^1	3^1	4^1	5^1	6^1	7^1				
Can Be			1^2	2^2	3^2	4^2	5^2	6^2	7^2		
Done At					1^3	2^3	3^3	4^3	5^3	6^3	7^3
That Time							1^4	2^4	3^4	4^4	5^4
									1^5	2^5	3^5
											1^6

Table 1. Parallelism of SOR

i^ℓ denotes the ℓ^{th} improvement of the elements in set i.

In Table 1 we see that steps 1 to 5 improve less than 8 elements but 8 elements are improved at each time step following step 5. Thus if we

have a computer with 8 PEs we can compute $U^{(\ell)}$ in $5 + 2\ell$ time steps. If we assume that our initial guess is 1^3, 2^2, 3^2, 4^1, 5^1, 6^0, 7^0, instead of 1^0, 2^0, 3^0, 4^0, 5^0, 6^0, 7^0 then we can start at step 6 and do every iteration in 2 time steps.

This can be generalized for an arbitrary mesh as follows: First we group all the interior mesh points into two sets,

$$\text{ODD} = \{U_{i,j} \mid i + j \text{ is odd}\}, \qquad \text{EVEN} = \{U_{i,j} \mid i + j \text{ is even}\}$$

Time step 1 improves all the points in EVEN. Time Step 2 improves all the points in ODD. We will call this method modified successive over-relaxation (MSOR).

Remember all the points in ODD can be calculated using only points from EVEN and all the points in EVEN can be calculated using only points from ODD. Thus we can write a two part algorithm for MSOR [Young, Page 271].

$$U_{i,j}^{(\ell+1)} = \omega_{E\ell} \left(v \left(U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell)}\right) + h \left(U_{i-1,j}^{(\ell)} + U_{i+1,j}^{(\ell)}\right) - g\, \Psi_{i,j}\right) + (1-\omega_{E\ell})\, U_{i,j}^{(\ell)} \tag{18.1}$$

for all the points in EVEN

$$U_{i,j}^{(\ell+1)} = \omega_{D\ell} \left(v \left(U_{i,j+1}^{(\ell+1)} + U_{i,j-1}^{(\ell+1)}\right) + h \left(U_{i,j+1}^{(\ell+1)} + U_{i,j-1}^{(\ell+1)}\right) - g\, \Psi_{i,j}\right) + (1-\omega_{D\ell})\, U_{i,j}^{(\ell)} \tag{18.2}$$

for all the points in ODD.

In Young's discussion of MSOR, the optimal $\omega_{E\ell}$ and $\omega_{D\ell}$ for the ℓ^{th} iteration are calculated. We will limit our study to MSOR where $\omega_{E\ell} = \omega_{D\ell} = \omega_b$.

Intuitively one would think MSOR and SOR have very close rates of convergence. In fact, their "asymptotic rates of convergence" are equivalent.* If A is a convergent n x n complex matrix, for all ℓ sufficiently large, the average rate of convergence for ℓ iterations, $R(A^\ell)$, is $\lim_{\ell \to \infty} R(A^\ell) = -\ln\rho(A) \equiv R_\infty(A)$ [Varga, Page 67]. Thus for a sufficiently large ℓ, the asymptotic rate of convergence equals the average rate of convergence. We know if P is a permutation matrix that $\rho(PAP^{-1}) = \rho(A)$ [Birkhoff and Maclane, Page 249]. By multiplying (9) by P we get

$$PU^{(\ell+1)} = P(D-\omega E)^{-1}\{\omega F + (1-\omega)D\}P^{-1}PU^{(\ell)} \tag{19}$$

The asymptotic rate of convergence for (19) equals

$$-\text{Ln}\ \rho(P(D-\omega E)^{-1}\{\omega F + (1-\omega)D\}P^{-1}) = -\text{Ln}\rho((D-\omega E)^{-1}\{\omega F + (1-\omega)D\}) \tag{20}$$

which equals the asymptotic rate of convergence for SOR.

This not only shows that SOR and MSOR have the same asymptotic convergence rate, but that SOR applied to any permutation of U_I gives the same asymptotic rate

---

* The negative of the logarithm of the spectral radius of a convergent matrix A is the asymptotic rate of convergence, $R_\infty(A)$, for the matrix A [Varga, Page 67].

of convergence. Thus we can choose the ordering of U_I which best fits the machine being used.

2.3.3 SOR using Straight Storage

Now we shall examine the implementation of SOR on ILLIAC IV [Rudsinski]. We will compare two storage schemes and examine their effect on machine efficiency and their time per execution.*

The first storage scheme, Straight Storage, stores all the elements of row i before any of the elements of row j if $i < j$. The elements of each row appear in the order they are found in the mesh. The first element of each row is stored in PE 0 and the last element of the row is stored in PE I where I equals (n-1) mod 64, n is the number of columns. Figure 2 should help clarify this storage scheme.

How can we implement SOR using Straight Storage? We observe that the PEs which contain the odd elements of mesh row I contain the even elements of mesh row I + 1. This enables us to improve the odd elements of Rows I and I +1 simultaneously by using a PE integer index which accesses an element

---

* The timing method is explained in Appendix A. The terms used in timing are defined there also.

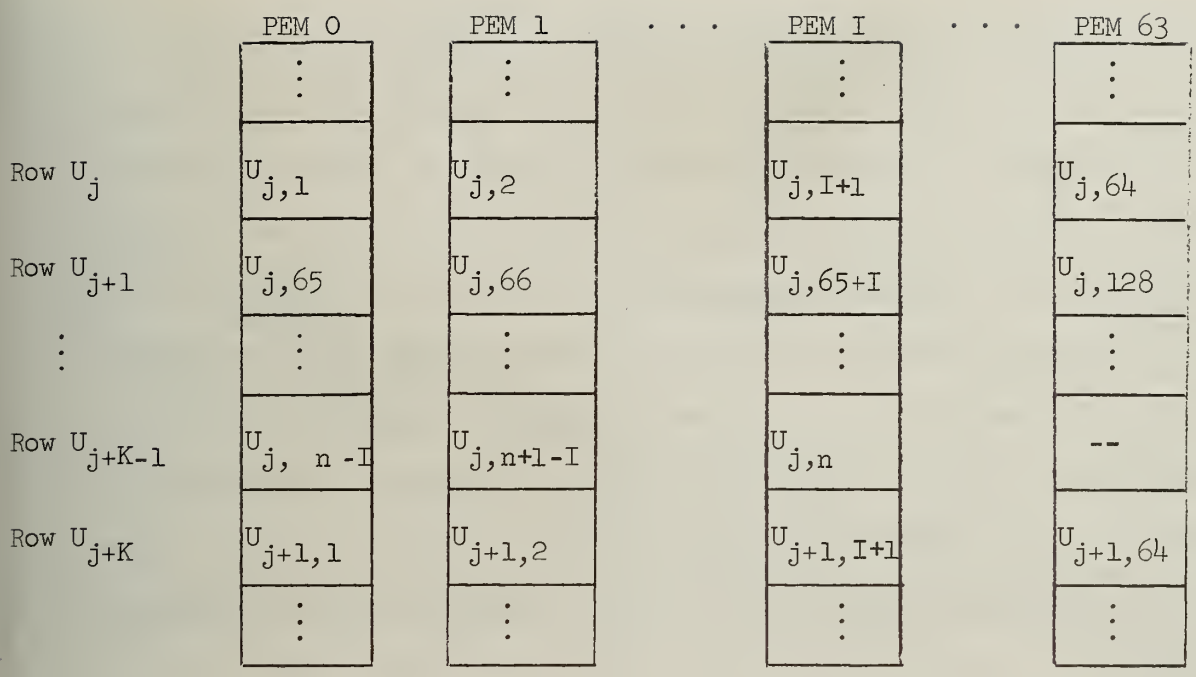


Figure 2. Straight Storage of Row j of U *

in row I (I+1) in the even (odd) PEs when I is even. In a like manner we can access all the even elements of two consecutive rows. If we have an odd number of rows we can improve the odd elements of the last row and the even elements of the first row simultaneously using a similar technique.

Given a mesh with m rows and n columns, we use the index described above so that we can improve two rows simultaneously. Each pair of rows can

---

* K stands for the number of rows of PEM required to store n words and I = (n-1) mod 64.

be divided into K groups where $K = \lfloor n/64 \rfloor$* if n mod 64 = 0 otherwise $K = \lfloor n/64 \rfloor + 1$. Using Phase 1 we can improve 64 elements at a time $\lfloor (n-2)/64 \rfloor$ times and then use Phase 2 to improve the remaining interior points.** If we use Phase 1 to improve the first 64 odd (even) interior points of a pair of lines then we must take into account that only 63 of them are found in the same pair of rows of PEM because of the boundary points. This means we must use special indices to reach the 64th odd (even) interior point. This requires no extra computer time because the number of times register X is changed is not affected. SOR requires not only the point to be improved but its four neighbors and the appropriate value of Ψ.

For example: To improve $U_{i,j}$ we need $U_{i-1,j}$, $U_{i+1,j}$, $U_{i,j+1}$, $U_{i,j-1}$, $U_{i,j}$ and $\psi_{i,j}$.

These values will be accessed by a combination of ACAR and Register X indexing. To do the indexing, K will be combined with the CU integer CI-equal to the first row of PEM containing mesh points to be improved during this execution - and three PE integers.

PI=(1,K,0,K,0,...,0,K), PR=(K,0,K,0,...,K,0,) and PL=(K+1,1,K,0,K,0,...,K,0) if we are improving the odd points;

PI=(K,1,0,K,0,K,...,K,0), PR=(0,K,0,K,...,0,K) and PL=(1,K+1,0,K,0,K,...,0,K) if we are improving the even points;

---

* $\lfloor p \rfloor$ equals k where p is a real number, k is an integer, and $k \leq p < k+1$.

** Phase 1 and Phase 2 are defined in Appendix D.

INITIAL CONDITIONS			
Register	Contents	Register	Contents
$ X	PI	$ D3	WI
$ DO	H	$ CO	CI
$ Dl	V	$ Cl	CM
$ D2	W	$ C2	CP

STEP NO.	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS
1	$A: = U[$X] (0)	PE fetch	7
2	$C3: = $D3; $A: = $A*$C3	CU fetch Real multiplication	10
3	$S: = $A - ψ[$X] (0)	Real subtraction (PE fetch overlapped by multiplication)	7
4	$A: = U[$X] (2)	(PE fetch overlapped by addition)	0
5	$A: = $A + U[$X] (1)	Real addition and PE fetch	14
6	$X: = PL	(PE fetch overlapped by addition)	0
7	$C3: = $DO; $A: = $A*$C3	CU fetch Real multiplication	10
8	$R: = U[$X] (0)	(PE fetch overlapped by multiplication)	0
9	$A: = $A + $S	Real addition	7
10	$X: = PR	(PE fetch overlapped by addition)	0
11	$B: = RTL (1,,$R)	Route	3
12	$R: = U[$X] (0)	(PE fetch partially overlapped by Route)	4
13	$S: = $A	Load from PE Register	1
14	$A: = RTR (1,,$R)	Route	3
15	$A: = $A + $B	Real addition	7
16	$C3: = $Dl; $A: = $A*$C3	(CU fetch overlapped by addition) Real multiplication	9
17	$X: = PI	(PE fetch overlapped by multiplication)	0
18	$A: = $A + $S	Real addition	7
19	$C3: = $D2; $A: = $A*$C3	(CU fetch overlapped by addition) Real multiplication	9
20	U[$X] (0): = $A	PE store	7
		TOTAL TIME	105

Table 2. Timing of an Execution of Phase 1 SOR using Straight Storage

Let $W = \omega g$, $V = v/g$, $H = h/g$, and $W1 = \frac{1-\omega}{W}$ where g, h, and v are defined in 1.2.

Now (17) can be written in GLYPNIR as follows:

```
CP: = C + K;
LOOP CI: = C, K + K, CK DO BEGIN
CM: = CI - K;
CP: = CI + K;
U [PI + CI]: = W*(V*(RTR(1,,U[PR+CI])+ RTL(1,,U[PL+CI])) + H*(U[PI+CM]
+ U[PI+CP]) - ψ[PI+CI] + W1*U[PI+CI] );                          (21)
END;
```

By placing CI in ACAR0, CM in ACAR1, CP in ACAR2, H in $D0, V in $D1, W in $D2, W1 in $D3 and PI in $X (21) could be translated into ASK as outlined in Table 2.*

2.3.4 SOR using Odd-Even Storage

Now we will look at another storage scheme to see if we can improve the data access time. Given an m by n mesh divide each row into k + 1 segments such that k segments are 128 mesh points long and the last segment is ℓ points long, where $0 \leq \ell < 128$. Use two adjacent lines of PEM to store each segment. Store all the red (black) points** of the segment in the first (second) line of PEM. The segments appear in the same relative order as they appeared in Straight Storage. This storage scheme will be called Odd-Even Storage. For an example see Figure 3.

---

* Appendix A explains the notation and timing method used in Table 4.

** A point that appears in an odd(even) numbered column is considered a red (black) point.

If $\ell \leq 64$, then Straight Storage requires $m\lceil \frac{n}{64} \rceil$ * lines of PEM while Odd-Even Storage required $m(\lceil \frac{n}{64} \rceil + 1)$ lines of PEM. If $\ell > 64$ then both Straight Storage and Odd-Even Storage require $m\lceil \frac{n}{64} \rceil$ lines of PEM.

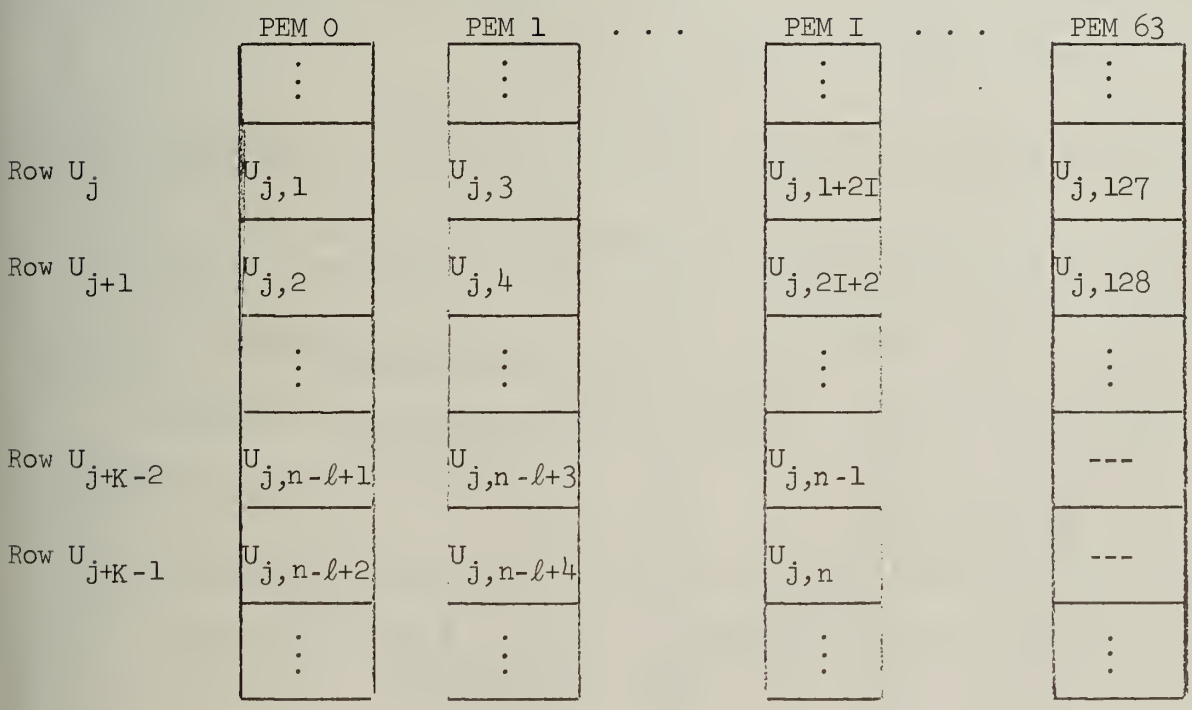


Figure 3. Odd-Even Storage Data Allocation of Row j of U Where n is Even.

---

* $\lceil p \rceil = k$ where k is an integer such that $p \leq k < p + 1$.

INITIAL CONDITIONS			
Register	Contents	Register	Contents
$ X	PI	$ D3	W1
$ D0	H	$ C0	CI
$ D1	V	$ C1	CM
$ D2	W	$ C2	CP

STEP NO.	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS
1	$A: = U[$X] (0)	PE fetch	7
2	$C3: = $D3; $A: = $A*$C3	CU fetch Real multiplication	10
3	$S: = $A - ψ[$X] (0)	(PE fetch overlapped with mult.) Real subtraction	7
4	$C3: = $C0 + 1	(CU operations overlapped)	0
5	$R: = RTL (1,,U(3))	(PE fetch overlapped by addition) Route	3
6	$A: = $R + U[$X](3)	(PE fetch partially overlapped by Route) Real addition	11
7	$C3: = $D1; $A: = $A*$C3	(CU fetch overlapped by addition) Real multiplication	9
8	$R: = U[$X](1)	(PE fetch overlapped by multiplication)	0
9	$S: = $A + $S	Real addition	7
10	$A: = $U[$X] (2) + $R	(PE fetch overlapped by addition) Real addition	7
11	$C3: = $D0; $A: = $A*$C3	(CU fetch overlapped by addition) Real multiplication	9
12	$A: = $A + $S	Real addition	7
13	$C3: = $D2; $A: = $A*$C3	(CU fetch overlapped by addition) Real multiplication	9
14	U[$X] (0): = $A	PE store	7
		TOTAL TIME	93

Table 3. Timing of an Execution of Phase 1 SOR using Odd-Even Storage

Phase 1 of the implementation of SOR using Odd-Even Storage requires only one PE index and one route. We can access the data using PI = (2,0,0,0..,0) and CI equal the first row of PEM containing mesh points to be improved during this execution. We need two assignment statements: (22.1) for the red points, and (22.2) for the black points.

Using K, W, V, H and W1 are defined in 2.2.3, SOR using Odd-Even Storage can be written in GLYPNIR as follows:

```
LOOP CI: = C, K, CK  DO BEGIN
CM: = CI - K;
CP: = CI + K;
U[PI + CI]: = W*(V*(RTR(1,, U[CI+1]) + U[PI + CI + 1])
 + H*(U[PI + CM] + U[PI + CP]) - Y[PI + CI] + WI*U[PI + CI]);    (22.1)
U[CI]: = W*(V*(RTL(1,,U[PI + CI - 1]) + U[CI - 1])
 + H*(U[CM] + U[CP]) - Y[CI] + W1*V[CI]);                        (22.2)
END;
```

Table 3 suggests how (22.1) could be coded in ASK. (22.2) requires different code but the logic is similar. The code for SOR using Odd-Even Storage is 10% faster than the code for SOR using Straight Storage. This is due to a savings in memory fetching and the elimination of one route.

2.3.5 SOR Machine Efficiency

If a mesh has m rows and 32 (64) columns and straight (odd-even) storage is used, machine efficiency can be doubled by placing rows 1 to $\frac{m}{2}$ in PE's 0 to 31 and rows $\frac{m}{2}$ + 1 to m in PE's 31 to 63. This improved storage scheme requires that rows $\frac{m}{2}$ and $\frac{m}{2}$ + 1 be handled as special cases. The idea behind this technique can be used on any mesh which doesn't have a multiple

of 64 (128) columns when straight (odd-even) storage is used. Implementation of such techniques increases machine efficiency and decreases storage requirements per mesh, but also increases the complexity of the program.

2.4 Implementation of SLOR on ILLIAC IV

2.4.1 Parallelism in SLOR

In 2.3.2 the scheme for SOR was altered to maximize parallelism on ILLIAC IV. In a similar manner, we will modify the scheme for SLOR to maximize the number of columns which can be improved simultaneously. In 2.4.2 the need for blocks of 128 columns to perform column relaxation with optimal efficiency on ILLIAC IV will be explained. This leaves no restrictions on the number of rows other than the limits of available core. Analogous to the SOR scheme, note that column i can be improved for the second time after column i + 1 is improved for the first time. By taking advantage of this fact we can improve half of the columns simultaneously after a finite number of iterations. Thus once we reach that state we can improve the entire mesh in two steps if we have enough PEs. In SOR we improve the odd (even) points and then the even (odd) points. In SLOR we must improve the odd (even) columns and then the even (odd) columns. The proof we used in 2.1.2 to show that we could improve the points in SOR in any order can be extended to show that the order in which the columns are improved SLOR does not change the asymptotic rate of convergence.

$$U_{i,j}^{(\ell+1)} = h(U_{i+1,j}^{(\ell+1)} + U_{i-1,j}^{(\ell+1)}) + \omega v(U_{i,j+1}^{(\ell)} + U_{i,j-1}^{(\ell)}) - \omega g \Psi_{i,j} \qquad (23.1)$$
$$+ (1 - \omega)(U_{i,j}^{(\ell)} - h(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell)}))$$

for the odd columns and

$$U_{i,j}^{(\ell+1)} = h(U_{i+1,j}^{(\ell+1)} + U_{i-1,j}^{(\ell+1)}) + \omega v(U_{i,j+1}^{(\ell+1)} + U_{i,j-1}^{(\ell+1)}) - \omega g \Psi_{i,j} \qquad (23.2)$$
$$+ (1 - \omega)(U_{i,j}^{(\ell)} - h(U_{i+1,j}^{(\ell)} + U_{i-1,j}^{(\ell)}))$$

for the even columns.

2.4.2 Storage for SLOR

Now we need to find a storage scheme which is efficient on ILLIAC IV. If we use Straight Storage, we run into the same inefficiency we did with that storage in implementing SOR. We need to use three PE integer indices and do two routes per mesh point. Furthermore, the method works on blocks of 256 columns. Column J and column J + 1 cannot be improved simultaneously. Thus we improve the odd (even) columns in the set $\{J \mid I \leq J \leq I + 63\}$ and the even (odd) columns in the set $\{J \mid I + 128 \leq J \leq I + 191\}$. If we use Odd-Even Storage we can eliminate one route and two of the PE indices. We also decrease the blocks to 128 columns each.

2.4.3 Improving a Line by the Cuthill-Varga Method

The method of implementing SLOR must solve a tridiagonal matrix equation in each PE. Thomas' method [Todd, Page 395] is commonly used. A method by Cuthill and Varga can eliminate some of the work by doing some preconditioning if the tridiagonal matrix is positive definite and real symmetric. Even if the mesh spacings are not constant, Poisson's equation satisfies those conditions [Cuthill and Varga, Page 241].

We will apply the latter method, the Cuthill-Varga method, as follows. Because we are performing successive column over-relaxation we will use a permutation of A, PAP^{-1}, to discuss the technique. The equation under consideration will be $-PAP^{-1}PU_I = -PM$.

$$-PAP^{-1} = \begin{bmatrix} B, C & & & & \\ C, B, C & & & & \\ & C, B, C & & & \\ & & \ddots & \ddots & \\ & & & C, B, C & \\ & & & & C, B \end{bmatrix} \tag{24}$$

where

$$B = \begin{bmatrix} e, d & & & \\ d, e, d & & & \\ & \ddots & \ddots & \\ & & d, e, d & \\ & & & d, e \end{bmatrix}, \quad C = \frac{-1}{h_y^2} I, \quad e = +2\left(\frac{1}{h_y^2} + \frac{1}{h_x^2}\right), \quad d = \frac{-1}{h_x^2}.$$

B and C are m-2 by m-2 matrices. Let U_i and M_i denote the i^{th} column of PU and -PM respectively. Now using (24) we obtain

$$BU_i = M_i - C(U_{i+1} + U_{i-1}). \tag{25}$$

By assuming the left side of (25) is constant we are left with a tridiagonal system of equations. Because B satisfies the requirements for the Cuthill-Varga method, [Cuthill and Varga, Page 237], we can define a diagonal matrix

$$G = \begin{bmatrix} c_1 & & & \\ & c_2 & & \\ & & \ddots & \\ & & & c_{m-2} \end{bmatrix}$$

such that

$G^{-1}BG^{-1} = T'T,$

where*

$$T = \begin{bmatrix} 1 & t_1 & & & \\ & 1 & t_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & t_{m-3} \\ & & & & 1 \end{bmatrix} .$$

The elements of G and T are calculated as follows:

$$c_1 = e^{\frac{1}{2}}; \quad c_j = \{e - (\frac{d}{c_{j-1}})^2\}^{\frac{1}{2}}, \quad 2 \le j \le m-2, \tag{26}$$

and

$$t_j = \frac{d}{c_j \, c_{j+1}} \quad 1 \le j \le m-3. \tag{27}$$

By multiplying

$$BU_i = M_i - C(U_{i+1} + U_{i-1})$$

by G^{-1} we obtain

$$G^{-1}BG^{-1}GU_i = G^{-1}M_i - G^{-1}CG^{-1}G(U_{i+1} + U_{i-1}) .$$

Substituting Y_i for GU_i gives

$$G^{-1}BG^{-1}Y_i = G^{-1}M_i - G^{-1}CG^{-1} (Y_{i+1} + Y_{i-1})$$

or

$$T'TY_i = G^{-1}M_i - G^{-1}CG^{-1} (Y_{i+1} + Y_{i-1}) \tag{28}$$

Equation (28) is first solved for TY_i and then for Y_i.

The Cuthill-Varga method employs a two part algorithm to perform SLOR.

First Y_i^* is calculated using

$$T'TY_i^{*(\ell+1)} = G^{-1}M_i - G^{-1}CG^{-1}(Y_{i+1}^{(\ell)} + Y_{i-1}^{(\ell)}) \tag{29.1}$$

if i is odd and

---

* T' equals the transpose of T.

$$T' TY_i^{*(\ell+1)} = G^{-1}M_i - G^{-1}CG^{-1}\,(Y_{i+1}^{(\ell+1)} + Y_{i-1}^{(\ell+1)}) \qquad (29.2)$$

if i is even.

Then $Y_i^{*(\ell+1)}$ is combined with $Y_i^{(\ell)}$ to perform the over-relaxation,

$$Y_i^{(\ell+1)} = \omega\,[Y_i^{*(\ell+1)} - Y_i^{(\ell)}] + Y_i^{(\ell)} \quad . \qquad (30)$$

After all the iterations are completed, Y_i is converted to U_i using $G^{-1}Y_i = U_i$.

2.4.4 SLOR Implementation

Now let us discuss more specifically how we implement SLOR* to solve $\frac{\delta U}{\delta X^2} + \frac{\delta U}{\delta Y^2} = \Psi$. Odd-Even Storage issued for the reasons stated in Section 2.4.2. We use the Cuthill-Varga method and thus do the following preconditioning: calculate c_i, $1 \le i \le m-2$; t_i, $1 \le i \le m-3$; $\frac{1}{c_i}$, $1 \le i \le m-2$; and $\frac{1}{(h_y c_i)^2}$, $1 \le i \le m-2$. We use PEM storage for these values so that when we calculate c_i we can do 64 of the square roots simultaneously because the square root takes a great deal of time relative to other operations on ILLIAC IV. We use Grabone** to access the c_i, t_i, $\frac{1}{c_i}$ and $\frac{-1}{(h_y c_i)^2}$. We subtract the row boundary conditions times $\frac{1}{h_x^2}$ to the appropriate row of Ψ and then we multiply by G^{-1}. We multiply column boundary conditions by G to obtain Y_1 and Y_n so that they may be used in equations (29) and (30).

Now we are ready to do the iterations.*** Two phases are used. Phase 1 improves 64 columns simultaneously and Phase 2 improves the remainder. Phase 1 is used when the number of interior columns is greater than or equal to 128. Phase 1 starts at the left and works on groups of 128 interior colum

---

* We limit our program to handle an even number of columns.

** The GLYPNIR function is used to access a single PE value and send it to all PEs. Refer to Table 9.

*** The reader should be familiar with the material in Appendix A.

moving to the right until the number of remaining interior columns is less than 128. Since there are elements of 63 interior odd columns on the first line of PEM containing interior points we use PI and CI as defined in 2.3.4. m-2 rows of temporary storage, STORE, are used. We use two similar codes for Phase 1: one for the odd columns and one for the even columns. Phase 2 can use the same code as Phase 1 if the mode is changed at the appropriate time.

We will limit our discussion to the code for the odd columns, Phase 1. The rest of the code can be found in Appendix D.*

```
STORE[0]:= ψ[PI+CI]-GRABONE(DISB[0],0)*(RTR(1,,U[CI+1])+U[PI+CI+1]);
LOOP CJ:= 1,1,CM3 DO BEGIN C:= CJ.[16:42];
CI: = CI+K;                                                          (31)
STORE [CJ]:= ψ[PI+CI]-GRABONE(DISB[C],CJ)*(RTR(1,,U[CI+1])+
U[PI+CI+1])-GRABONE(E[C],CJ)*STORE[CJ-1];
END;
```

```
LOOP MC:=1,1,CM3 DO BEGIN
CJ:=CM3-MC;
C:=CJ.[16:42];
STORE[CJ]:=STORE[CJ]-GRABONE(E[C],CJ)*STORE[CJ+1];                   (32)
END;
```

---

* DISB contains $\frac{-1}{(h_y c_i)^2}$, Ψ contains $G^{-1}M$ and E contains t_i.

INITIAL CONDITIONS FOR FORWARD ELIMINATION

Register	Contents	Register	Contents
$A	STORE[CJ-1]	$C0	CI
$X	PI	$C1	CJ
		$C2	CI+1

STEP NO.	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS
1	$B: = GRABONE(E[C],CU)	Load	10
2	$A: = $A*$B	Real multiplication	9
3	$S: = ψ [$X](0)-$A	(PE fetch overlapped by mult.) Real subtraction	7
4	$R: = RTR(1,,U(2))	(PE fetch overlapped by subtract.) Route	3
5	$A: = U[$X](2)+$R	(PE fetch partially overlapped by route) Real addition	11
6	$R: = GRABONE(DISB[C],CU)	Load	10
7	$A: = $A*$R	Real multiplication	9
8	$A: = $S-$A	Real subtraction	7
9	STORE(1): = $A	PE store	7
		Subtotal	73

INITIAL CONDITIONS FOR BACKWARD ELIMINATION

Register	Contents	Register	Contents
$A	STORE[CJ+1]	$C0	CJ

STEP NO.	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS
1	$B: = GRABONE(E[C],CJ)	Load	10
2	$A: = $A*$B	Real multiplication	9
3	$A: = STORE(0)-$A	(PE fetch overlapped by mult.) Real subtraction	7
4	STORE(0): = $A	PE store	7
		Subtotal	33

INITIAL CONDITIONS FOR OVER-RELAXATION

Register	Contents	Register	Contents
$X	PI	$C1	CU
$C0	CI	$C2	W

STEP NO.	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS
1	$R: = U[$X](0)	PE fetch	7
2	$A: = STORE(1)-$R	PE fetch Real subtraction	14
3	$A: = $C2*$A	(CU fetch overlapped by sub.) Real multiplication	9
4	$A: = $A+$R	Real addition	7
5	U[$X](0): = $A	PE store	7
		Subtotal	44
		TOTAL TIME	150

Table 4. Timing of an Execution of Phase 1 SLOR

```
LOOP CJ:=0,1,CM3 DO BEGIN
ST:=U[PI+CI];
U[PI+CI]:=W*(STORE[CJ]-U[PI+CI])+U[PI+CI];
IF ABS(ST-U[PI+CI]) GRT BOUND THEN B:=1;% CHECKING THE ERROR BOUND (33)
CI:=CI+K;
END;
```

SLOR has three primary sections: (31) does the forward elimination, (32) does the backward elimination, and (33) performs the over-relaxation. In Table 4 we suggest how to code these sections efficiently in ASK and we give a time estimate for that code.

2.5 Implementation of ADI on ILLIAC IV

ADI improves all columns simultaneously and then improves all rows simultaneously. Thus, when we do the column (row) improvement we want the values of the boundary columns (rows), but the boundary rows (columns) could have been added to vector ψ in preconditioning. One solution is to include both the row and the column boundary values in each iteration. Another solution is to add them both in when preconditioning and then set the boundary values to zero so they will not affect the interior points when used in the iteration. If we need the values of the boundary after the problem has converged then we must store them before performing any of the iterations and replace them after all the iterations are completed.

2.5.1 Thomas' Method on ILLIAC IV

As in SLOR we have sets of tridiagonal matrices to solve. We used the Cuthill-Varga method in SLOR. This method does preconditioning to cut down on the computational time (see Section 2.4.3). ADI changes the diagonal at every iteration. Thus the preconditioning would have to be done at each interation. Thomas' Method [Todd, page 395] has the same problem but the preconditioning requires less computer time and the total computer time per iteration is smaller. Thus we will use Thomas' Method in solving the tridiagonal matrix problems. Instead of implementing the method in a straight forward manner, we use the fact that the horizontal coefficients and the vertical coefficients are constant.

The matrix problem* we want to solve is of the form

$$aT_1 + b\,T_2 = D_1$$

$$bT_{i-1} + aT_i + bT_{i+1} = D_i \quad (i = 2, 3, \ldots, n-3),$$

$$bT_{n-3} + aT_{n-2} = D_{n-2}$$

where

$$a = -\left(\frac{2}{h_y^2} + \omega_{\ell V}\right), \quad b = \frac{1}{h_y^2} \quad \text{for row relaxation}$$

and

$$a = -\left(\frac{2}{h_x^2} + \omega_{\ell H}\right), \quad b = \frac{1}{h_x^2} \quad \text{for column relaxation}$$

Thomas' method is as follows

$$e_1 = \frac{b}{a}, \quad e_i = \frac{b}{a - b\,e_{i-1}} \quad (i = 2, 3, \ldots, n-3), \tag{34}$$

$$q_1 = \frac{D_1}{a}, \quad q_i = \frac{D_i - bq_{i-1}}{a - b\,e_{i-1}} \quad (i = 2, 3, \ldots, n-2), \tag{35}$$

$$T_{n-2} = q_{n-2}, \quad T_i = q_i - e_i\,T_{i+1} \quad (i = n-3, n-4, \ldots, 1). \tag{36}$$

---

* We are considering an m by n mesh, U.

If we let $c = \frac{1}{b}$ we can rewrite the equation (35) as follows:

$$q_1 = e_1 \, c \, D_1, \quad q_i = e_i \, (c \, D_i - q_{i-1}) \; (i = 2,3, \ldots, n-2),$$

and $e_{n-2} = \frac{b}{a - b \, e_{n-3}}$. This reduces the number of divisions which are relatively time consuming on ILLIAC IV.

We have to calculate the e_i's and store them. For each row (column), they are the same. Calculating the e_i's is a serial process. The only way we could overlap is to calculate the set of e_i's for different iterations simultaneously. The problem with this solution is that it requires extra storage. If we calculate all the e_i's needed for 32 iterations we use max (m-2, n-2) rows of storage. If storage is available, then this is a feasible method; but if we must use disk I/O, the computational time saved will be overcome by increased I/O time.

2.5.2 Skewed Storage

The next problem we must overcome is how to access the rows on one sweep and then the columns on the next. Skewed storage was designed just for that purpose. We can place a m by n mesh in skewed Storage by performing the following algorithm:

1) Calculate the number of rows of PEM required to store n points. Set K equal to that number.

2) Place the m by n mesh in straight storage.

3) Route each of the K rows of PEM containing elements of mesh row i. Let the distance of the route be i-1.

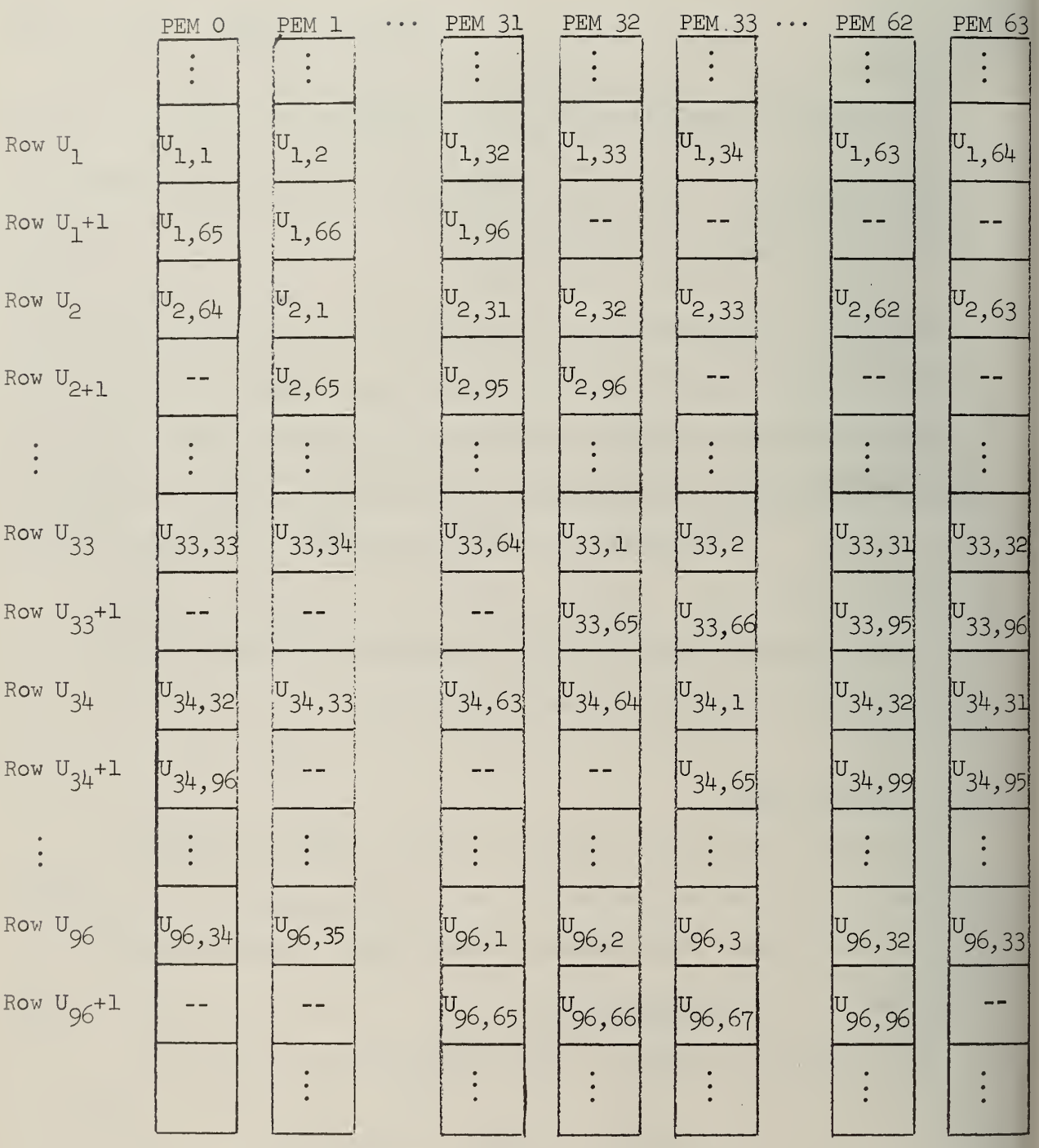


Figure 4. A 96 by 96 Mesh in Skewed Storage

ADI is a simultaneous method, (i.e. only values from the previous iteration are used, see (15.1) and (15.2)). On ILLIAC IV we can simultaneously improve up to 64 rows (columns). Thus if the dimensions of the mesh are less than 67 we can do all the rows (columns) in one time step. If one of the dimensions is greater than 66 then we need to use more than one time step for row or column relaxation. In this case we will use two phases to complete an iteration. An m by n mesh will be divided using $R = \left\lfloor \frac{m-2}{64} \right\rfloor$ and $J = \left\lfloor \frac{n-2}{64} \right\rfloor$. Phase 1 row (column) relaxation will improve R (J) groups of 64 consecutive rows (columns). Then Phase 2 will improve the remaining interior rows (columns). Phase 1 starts with row (column) 2 and works toward row (column) m-1 (n-1). When Phase 1 improves the I^{th} group where $1 \leq I \leq R$ (J) and k is the first row (column) of group I, the old values of row (column) k-1 are needed. Thus to improve group I Phase 1 goes through the following steps:

1) The values of row (column) k-1 are placed in the temporary storage, NEW.

2) The values in the temporary storage, OLD are placed in the PEM storage for row (column) k-1.

3) The values or row (column) k+63 are placed in the temporary storage, OLD.

4) Rows (columns) k through k+63 are improved.

5) The values in NEW are placed in row (column) k-1.

When Phase 1 is improving the first group steps 1, 2 and 5 can be omitted. Phase 2 needs to do steps 1, 2, 4 and 5.

Now we will discuss the type of indexing used in accessing the mesh points required to perform Phase I column relaxation using skewed storage. We access* the set $\{(\psi_{i,j}, U_{i,j}) | k \leq j \leq k + 63\}$ by using a permutation of the PE integer, (1,0,0, $\cdots$,0). PI will stand for that permutation. PI will be used as a PE index and CI will be the CU index. PI must be routed one PE to the right to access the set $\{U_{i+1,j} | k \leq j \leq k+63\}$. PI must be routed one PE to the left to access $\{U_{i-1,j} | k \leq j \leq k+63\}$. To access the set $\{U_{i,j+1} | k \leq j \leq k+63\}$ we must add RTR(1,,PI) to PI. The set $\{U_{i,j-1} | k \leq j \leq k+63\}$ can be accessed using only CI. After being accessed the four neighbors of $U_{i,j}$ must be routed to the PE containing $U_{i,j}$ before $U_{i,j}$ can be improved. The following GLYPNIR code uses this method of data access to perform Phase 1, Step 4 of column relaxation:

```
LOOP C:=1,1,N-2 DO BEGIN CC:= C. [16:42]; CI:=CI+K;
U[PI+CI]:=(AI*(ψ [PI+CI]-BETA*(RTL(1,,U[PI+RPI+CI])+RTR(1,,U[CI]))
+CA*U[PI+CI])-RTR(1,,U[LPI+CI-K]))*GRABONE(B[C-1],CC);                (37)**
LPI:=PI;PI:=RPI;RPI:=RTR(1,,RPI);END;

LOOP IC:=0,1,N-3 DO BEGIN C:=N-2-IC;
CC:+C.[16:42]; CI:=CI-K;
U[PI+CI]:=U[PI+CI]-GRABONE(B[C-1],CC)*RTL(1,,U[RPI+CI+K]);           (38)
RPI:=PI;  PI:=RTL(1,,PI);END;
```

---

* The variable, k equals the first column of the group being improved.

** $AI=-h_y^2$; $BETA= 1/h_x^2$;W equals the relaxation parameter for this iteration; CA=2*BETA+W.

The code is divided into two sections: Section (37) does Thomas' forward elimination. Section (38) does Thomas' backward elimination; Table 7 suggests how this code could be translated into ASK and how much time the ASK code would take.

By replacing PI with KPEN=RTR(I,,PENK) where PENK=(0,K,2k,...,63K) and K is the number of lines of PEM required to store a row of the mesh, we would have Phase 1 row relaxation.*

A complete program for a m by n mesh where n, $m \leq 64$ is found in Appendix E.

---

* In row relaxation a PE index must be used to access $\{U_{i,j-1} | k \leq j \leq k+63\}$.

INITIAL CONDITIONS FOR FORWARD ELIMINATION

Register	Contents	Register	Contents
$C1	CI	$D2	CA
$DO	AI	$X	PI
$D1	BETA	$C3	CI-K

STEP NO.	ASSIGNMENT STATEMENT	OPERATIONS AND OVERLAP	TIME IN CLOCKS
1	$A: = U[$X](1)	PE fetch	7
2	$C2: = $D2;$A:=$A*$C2	CU fetch and real mult.	10
3	$S: = $A + Ψ[$X](1)	(PE fetch overlapped by mult.) Real addition	7
4	PI: = $X	(PE store overlapped by add.)	0
5	$X: = $X + RPI	PE fetch and integer addition	10
6	$B: = RTL(1,,U[$X](1))	(PE fetch partially overlapped by addition) Route	7
7	$A: = RTR(1,,U(1))	(PE fetch partially overlapped by route) Route	7
8	$X: = LPI	(PE fetch partially overlapped	4
9	$A: = $A + $B	Real addition	7
10	$R: = U[$X](3)	(PE fetch overlapped by add.)	0
11	$C2: = $D1;$A:=$A*$C2	CU fetch, real multiplication	10
12	$X: = PI	(PE fetch overlapped by mult.)	0
13	$A: = $S-$A	Real subtraction	7
14	LPI: = $X	(PE store overlapped by subt.)	0
15	$C2: = $DO;$A:=$A*$C2	CU fetch, real multiplication	10
16	$B: = RTR(1,,$R)	Route	3
17	$A: = $A+$B	Real addition	7
18	$S: = RPT	(PE fetch overlapped by add.)	0
19	$R: = RTR(1,,$S)	Route	3
20	$A: = $A*GRABONE(B[CI-1],CC)	Load real multiplication	19
21	RPI: = $R	(PE store overlapped by mult.)	0
22	U[$X](1): = $A	PE store	7
23	$X: = $S	PE register to PE register load	1
		Subtotal	126

INITIAL CONDITIONS FOR BACKWARD ELIMINATION

Register	Contents	Register	Contents	Register	Contents
$X	PI	$A	U[RPI+CI+K]	$C1	CI

STEP NO.	ASSIGNMENT STATEMENT	OPERATIONS AND OVERLAP	TIME IN CLOCKS
1	$B: = GRABONE(B[C-1],CC)	Load	10
2	$A: = $B*$RTL(1,,$A)	Route, real multiplication	12
3	$A: = U[$X](1)-$A	(PE fetch overlapped by mult.) Real subtraction	7
4	U[$X](1): = $A	PE store	7
5	$X: = RTL(1,,$X)	(Route overlapped by store)	0
		Subtotal	36
		TOTAL TIME	162

Table 5. Timing of an Execution of Phase 1 ADI

2.6 Summary of the Results for Iterative Methods

Now that we have examined implementation of the various methods we can make some recommendations concerning the use of the methods. Table 6 contains a numerical study of convergence rates of the method applied to Poisson's equation where $h_x = h_y = \frac{1}{15}$ on different mesh sizes. The error bound was 0.0001.

Method \ Mesh size		64x64	32x64	32x32	16x64
SLOR Column Relaxation	ω	1.86415	1.79300	1.74760	1.64750
	I_t	63	40	32	21
	I_a	66	41	33	23
SOR	ω	1.90645	1.85610	1.82147	1.75050
	I_t	92	59	41	32
	I_a	90	63	44	28
ADI	I_a	10	9	8	7
	η	9	8	7	6

a) ω is the relaxation parameter

b) I_t is the theoretical number of iterations

c) I_a is the actual number of iterations*

d) 2η is the number of relaxation parameters used in ADI

Table 6. A Numerical Study of the Methods under Discussion

We can see from Table 6 that ADI is consistently the fastest**, then comes SLOR, with SOR being slowest. This conforms with theoretical results. In fact, in a square mesh where $h_x = h_y$, SLOR converges $\sqrt{2}$ times faster than SOR while ADI converges β times faster than SOR. β is a monotonically increasing function of the number of mesh points [Todd, Pages 396-398].

---

* Remember ADI improves the points twice in each iteration.

** The speed of the algorithms is compared with respect to the number of iterations.

To compare the time required to perform one of the above methods on ILLIAC IV to that required to perform the method on a specific machine we need to calculate the clocks per point per iteration and compare it with the clocks per point per iteration on ILLIAC IV. We have shown that SOR, SLOR, and ADI can be programmed to enable ILLIAC IV to improve 64 points simultaneously for certain mesh sizes.* By dividing the clocks required to improve 64 points simultaneously by 64 we obtain the clocks per point per iteration on ILLIAC IV. Table 7 contains these values and other pertinent information.

		ADI	SLOR	SOR
Number of operations per point per iteration	Mult.	10	4	4
	Add.	10	6	5
Time in clocks per 64 points per iteration	Arithmetic	160	78	71
	Data Access	164	72	22
Total clocks per point per iteration		5.06	2.34	1.45
Optimum mesh size**		64I by 64L	128I by L	64I by 2L
Rows of PEM required for temporary storage		Max (n-2,m-2)	$m-2+4\frac{(m-2)}{64}$	0

Table 7. Summary of Time and Storage Requirements for the Iterative Methods on a m by n Mesh.

Most of the data access time on ILLIAC IV could be eliminated if it was a serial machine. Let Machine A be a serial machine with a processor similar in speed to a PE*** but with the ability to perform ADI, SOR, and SLOR with negligible data access time.**** From Table 7 we can see that Machine A would take 71/1.45, about 48, times longer to perform SOR than ILLIAC IV would take. Machine A would take about 32 times longer than ILLIAC IV to perform ADI or SLOR.

---

* For maximum efficiency on ILLIAC IV, one dimension of the mesh must be a multiple of 64 for SOR and SLOR while both dimensions must be multiples of 64 for ADI.

** I and L are positive integers

*** A single PE has approximately twice the arithmetic speed of a CDC 6600.

**** The clocks per point per iteration on Machine A equal the arithmetic time in clocks per 64 points per iteration on ILLIAC IV.

The times in Table 7 are assuming the inner loops of the programs are written in ASK. If the programs were written completely in GLYPNIR the times would about double.

Tables 6 and 7 indicate that ADI is the fastest iterative algorithm on ILLIAC IV for rectangular meshes. The theory for ADI has only been developed for special cases [Young, Page 555]. If we have a commutative case, HV = VH, then ADI will be effective. The commutative space requires a rectangular region and coefficients of the differential equation which are sufficiently regular [Young, Page 538]. In some noncommutative cases, numerical experiments have shown ADI to converge rapidly. This is not always true. In some noncommutative cases ADI fails to converge for certain parameters [Young, Page 546]. More work needs to be done on noncommutative cases before ADI can be used freely on them.

ADI performs row relaxation followed by column relaxation. Thus to efficiently perform ADI on a m by n mesh on ILLIAC IV, both m and n must be divisible by 64. For example if we have a 16 by 64 mesh when we perform row relaxation, ILLIAC IV will be working at 25% machine efficiency while for column relaxation ILLIAC IV would be working at 100% machine efficiency. If one had four 16 by 64 meshes to solve, ILLIAC IV could solve the four meshes simultaneously and thus brings the machine efficiency up to 100%.

SLOR converges in fewer iterations than SOR and can be programmed to take about the same amount of time per iteration as SOR on most serial machines. On ILLIAC IV SLOR requires data from one PE broadcast to all the PEs while SOR does not. This is the major factor in causing SLOR to

take about 1.5 times longer per iteration than SOR, see Table 7. Thus SOR is a faster method on ILLIAC IV unless it takes at least 1.5 times more iterations to converge. This is rarely the case. SLOR requires considerably more temporary storage than SOR. Finally SLOR is a block iterative method and thus cannot be as easily reformulated as SOR for efficient performance on ILLIAC IV.

As shown in Section 2.3 of this study, SOR has a few constraints on how it must be implemented. This enables one to program it efficiently on ILLIAC IV for most mesh geometrics. SOR seems to be the most promising of the iterative and direct methods examined by the author for non-rectangular meshes.

In this study, we indicated that the order in which the elements are improved by SOR does not affect the asymptotic rate of convergence. This can be misunderstood. The number of iterations required to obtain a specified error bound is dependent on the order in which the elements are improved. For specific initial conditions, one ordering might give a faster initial rate of convergence than another. In some applications of Poisson's equation the actual number of iterations required for convergence is substantially smaller than theoretically expected because a good initial guess is supplied. In these cases the ordering has a greater affect on the computer time required to get an acceptable solution. ADI and SLOR require a considerable amount of preconditioning before the iterative process can begin. Thus if the initial guess is good enough, SOR will take the least amount of computer time on any mesh.

This study examined straight and odd-even storage for SOR. The use of odd-even storage improved the speed of SOR by 10%. In most problems, straight storage is used. Thus, for a valid comparison of the two storage schemes, the time to convert straight storage to odd-even storage must be considered. This takes about the same amount of time as one SOR iteration. Thus the program must require at least 20 iterations to justify converting to odd-even storage and then back to straight storage.

3. IMPLEMENTATION OF DIRECT METHODS ON ILLIAC IV

3.1 Introduction to Hockney's Direct Method (FACR)

Direct methods for solution of a restricted class of Poisson's equation have been developed which are faster than any iterative method developed to date [Dorr, Pages 258-259]. To the author's knowledge, Hockney's Fourier Analysis/Cyclic Reduction (FACR) is the fastest direct method on serial machines [Hockney, Page 159]. In this paper we will examine FACR and modify it so that it can be programmed efficiently on ILLIAC IV. FACR solves the "five-point" difference formula on a rectangular mesh; namely,

$$\frac{U_{s-1,t} - 2U_{s,t} + U_{s+1,t}}{h_x^2} + \frac{U_{s,t-1} - 2U_{s,t} + U_{s,t+1}}{h_y^2} = \Psi_{s,t} \qquad (39)$$

where the number of points being computed in one direction is $2^N - 1$ for Dirichlet's boundary conditions, 2^N for periodic boundary conditions and $2^N + 1$ for Neumann's boundary conditions. These three boundary conditions are permitted in the x and y direction, giving nine possible combinations of boundary conditions.

A 2^m by 2^n mesh produces a linear system with 2^{n+m} unknowns. Let $M = 2^m$ and $N = 2^n$. FACR solves this system using the following five step algorithm:

1. Given Ψ compute Ψ^* for the even columns and overwrite Ψ with Ψ^* on the even columns.
2. Using Ψ^* compute Ψ^s and overwrite Ψ^* with Ψ^s.
3. Using Ψ^s solve for U^s and overwrite Ψ^s with U^s.
4. Using U^s compute U on the even columns and overwrite U^s with U on the even columns.

5. Using the values of U on the even columns solve (39) for the values of U on the odd columns and overwrite Ψ with U on the odd columns.

The superscripts are for Dirichlet's boundary conditions and are defined as follows:

$$\Psi^*_{i,j} = \Psi_{i,j+1} - \frac{h_y^2}{h_x^2}\left(\Psi_{i-1,j} + \Psi_{i+1,j}\right) + 2\left(\frac{h_y^2}{h_x^2} + 1\right)\Psi_{i,j} + \Psi_{i,j-1}$$

$$\Psi^s_{i,j} = \frac{2}{M}\sum_{k=1}^{M-1} \Psi^*_{k,j} \sin\frac{\pi ik}{M}$$

$$U^s_{i,j} = \frac{2}{M}\sum_{k=1}^{M-1} U_{k,j} \sin\frac{\pi ik}{M}$$

Odd/even reduction divides the problem into two parts; first to solve a linear system concerning even columns and secondly to solve for the values on the odd columns. The even columns form a linear system with $M\frac{N}{2}$ unknowns which is solved by steps 2, 3, and 4. Step 5 involves the $\frac{N}{2}$ linear systems with M unknowns that give the values for the odd columns.

Fourier analysis decouples the even columns into M tridiagonal linear systems with $\frac{N}{2}$ unknowns. The unknowns are the Fourier harmonics, U^s, of U.

Recursive cyclic reduction solves the linear systems for the Fourier harmonics U^s.

Fourier synthesis converts the Fourier harmonics U^s obtained by the previous step to U giving the values of U on the even columns.

The values of U on the even columns are used to calculate the values of U on the odd columns. This involves solving $\frac{N}{2}$ tridiagonal linear systems with M unknowns [Hockney, Pages 148-153].

Refer to Appendix B for a more detailed explanation. Note that all five steps can be performed using one mesh containing the boundary conditions and Ψ initially. This is overwritten by Ψ^*, Ψ^s, U^s and finally the solution U.

We will discuss the method in three parts, steps 1 and 5, step 3, and steps 2 and 4. Suggestions will be made for the application of each part. Finally we will present a program in GLYPNIR similar to FACR and make suggestions on how it can be programmed for ASK.

3.2 Odd/Even Reduction and Odd Column Solution

Odd/even reduction is used to cut down on the number of columns where Fourier analysis and synthesis is applied. Consider the three neighboring equations:

$$\begin{aligned} \frac{1}{h_y^2} U_{t-2} + BU_{t-1} + \frac{1}{h_y^2} U_t &= \Psi_{t-1} \\ \frac{1}{h_y^2} U_{t-1} + BU_t + \frac{1}{h_y^2} U_{t+1} &= \Psi_t \\ \frac{1}{h_y^2} U_t + BU_{t+1} + \frac{1}{h_y^2} U_{t+2} &= \Psi_{t+1} \end{aligned} \qquad (40)$$

where U_t is the t^{th} column of the mesh and t is even. By multiplying the middle even line equation by $-Bh_y^2$ and adding we obtain

$$\frac{1}{h_y^2} U_{t-2} + \left(\frac{2}{h_y^2} - B^2 h_y^2\right) U_t + \frac{1}{h_y^2} U_{t+2} = \Psi_{t+1} - Bh_y^2 \Psi_t + \Psi_{t-1} = \Psi_t^* \qquad (41)$$

Note equation (41) contains only even columns of U, that is U_{t-2}, U_t and U_{t+2}. Thus after (41) is solved the algorithm must solve for the values of U on the odd columns.

To apply odd/even reduction one needs an odd number of mesh points for Dirichlet or Neumann boundary conditions and an even number of mesh points for periodic boundary conditions.

Odd/even reduction halves the number of columns that require Fourier analysis and synthesis. The odd columns are solved as tridiagonal matrix problems, step 5. Cyclic reduction is used to solve for the odd columns of U. This has the potential of saving computational time. Hockney showed that one level of odd/even reduction reduces the total number of operations per point on a serial machine from 34 to 24 on a 128 x 128 mesh [Hockney, Page 161].

Let k be the number of interior columns in the mesh. On ILLIAC IV Fourier analysis and synthesis can be applied to 64 of these columns simultaneously. Thus without odd/even reduction a Fourier algorithm must be applied $\lceil \frac{k}{64} \rceil$ times.* If odd/even reduction is used a Fourier algorithm must be applied $\lceil \frac{k}{128} \rceil$ times.

Odd/even reduction is of no use if $k \leq 64$ because then $\lceil \frac{k}{64} \rceil = \lceil \frac{k}{128} \rceil$. If $k > 64$ then $\lceil \frac{k}{64} \rceil > \lceil \frac{k}{128} \rceil$ and odd/even reduction would reduce the number of times Fourier analysis and synthesis must be applied. The best case is where $\lceil \frac{k}{64} \rceil = 2 \lceil \frac{k}{128} \rceil$ in which case the number of Fourier analysis and synthesis is cut in half.

Straight storage is the standard storage scheme used by most programs. If odd/even reduction is used, maximum machine efficiency is obtained when an even column is contained in each PE for the odd/even

---

* $\lceil P \rceil$ equals the integer L where $L - 1 < P \leq L$.

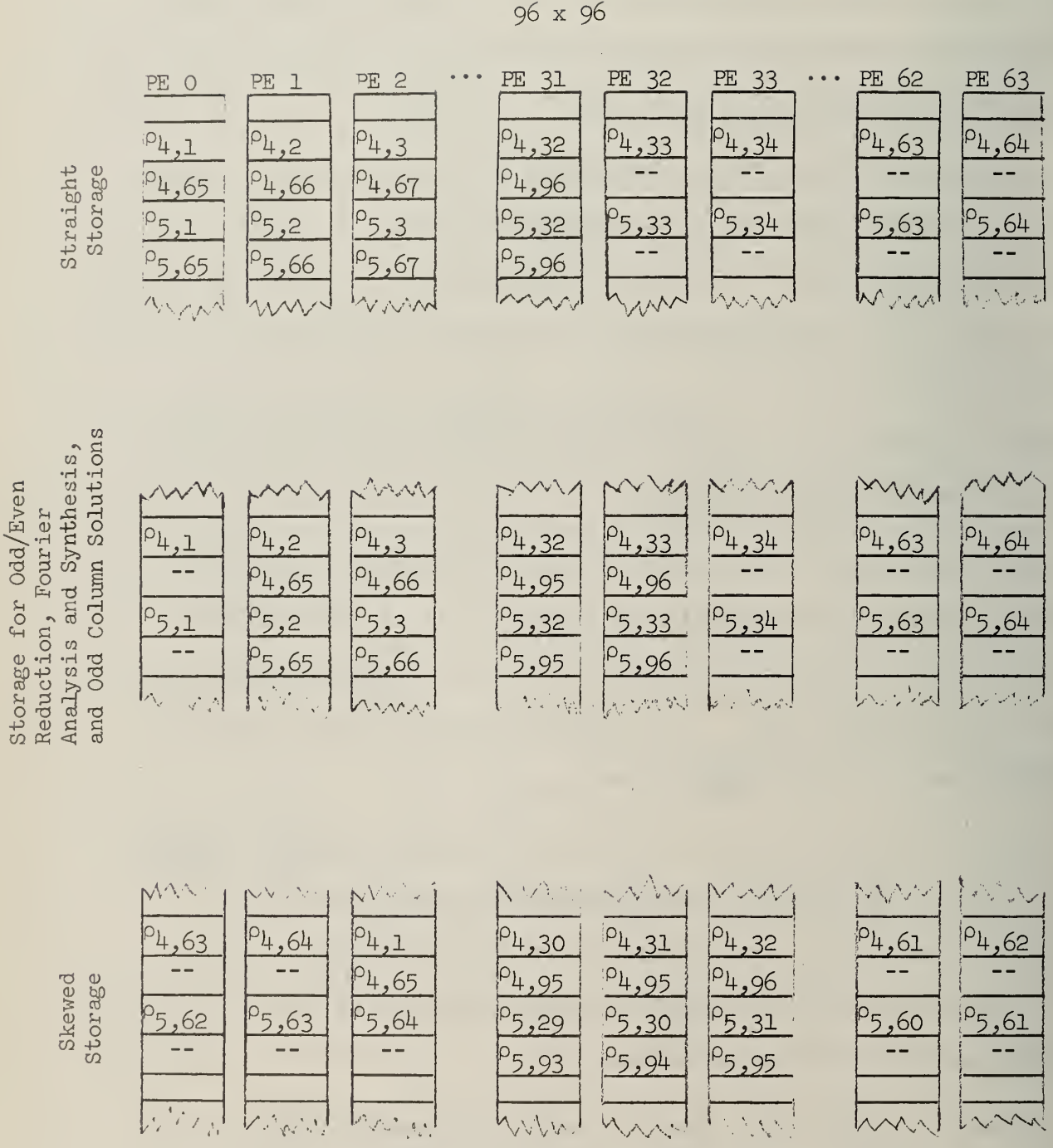


Figure 5. Storage Schemes Used on Rows 4 and 5 of a 96 by 96 Mesh if Odd/Even Reduction is Used

reduction step, the Fourier analysis step and the Fourier synthesis step; CRED can access a different row in each PE; and during the solution of the odd columns an odd column can be accessed by each PE. The following storage changes will fulfill all these conditions assuming each row of the mesh requires an even number of rows of PEM.

1. We assume the main program supplies U in straight storage.
2. Before odd/even reduction, the even rows of PEM containing U are routed one PU to the right of the odd rows.
3. Before CRED, U is placed in skewed storage.
4. Before Fourier synthesis, U is returned to the storage used for odd/even reduction.
5. The mesh U is placed in straight storage at the end of the subroutine.

See Figure 5 for clarification of the above storage schemes. If odd/even reduction is not used steps 2 and 4 of the storage changes need not be executed. The use of odd/even reduction requires the extra storage changes because steps one through four of FACR are applied to even columns while step five solves the odd columns. Without the use of odd/even reduction steps 2, 3, and 4 in Section 3.1 would be applied to all the columns.

3.3 CRED

In step 3, section 3.1 we have a tridiagonal system of equations to solve. The derivation of the coefficients for Dirichlet's boundary conditions is contained in Appendix B. By multiplying equation (B-8) in Appendix B by h_y^2 the diagonal coefficient for the k^{th} row becomes

$$\lambda_k = -\left(2\,\frac{h_y^4}{h_x^4}\cos\frac{2\pi k}{M} - 8\left(\frac{h_y^4}{h_x^4} + \frac{h_y^2}{h_x^2}\right)\cos\frac{\pi k}{M} + 6\,\frac{h_y^4}{h_x^4} + 8\,\frac{h_y^2}{h_x^2} + 2\right)$$

Hockney solves these tridiagonal systems using cyclic reduction recursively. Given

$$\begin{aligned} U^s_{k,t-4} + \lambda_k U^s_{k,t-2} + U^s_{k,t} &= h_y^2\,\psi^s_{k,t-1} \\ U^s_{k,t-2} + \lambda_k U^s_{k,t} + U^s_{k,t+2} &= h_y^2\,\psi^s_{k,t} \qquad (42) \\ U^s_{k,t} + \lambda_k U^s_{k,t+2} + U^s_{k,t+4} &= h_y^2\,\psi^s_{k,t+1} \end{aligned}$$

where t is even.

By multiplying the middle equation by $-\lambda_k$ and adding we obtain

$$U^s_{k,t-4} + (2 - \lambda_k^2)\,U^s_{k,t} + U^s_{k,t+4} = h_y^2\,(\psi_{k,t-2} + \psi_{k,t+2} - \lambda\,\psi^s_{k,t}) \qquad (43)$$

Now the process is repeated on the equations until we are left with just one equation. Then the process is reversed [Hockney, Page 150]. This method uses $\mathrm{Log}_2 N$ extra words of storage, $4N - 2\mathrm{Log}_2 N$ additions, $2N$ multiplications and $\mathrm{Log}_2 N$ divisions for an N variable system.* It also requires that N equal $2^I - 1$ if the system has Dirichlet conditions, 2^I if the system has periodic conditions, and $2^I + 1$ if the system has Neumann conditions.

For Dirichlet boundary conditions Thomas' method allows one to solve a tridiagonal system of N variables where there are no restrictions on N, (See Section 2.5.1). In the case where the diagonal elements are equal and all the other non-zero elements equal 1, we calculate

---

* N is the number of interior even columns in the mesh.

$$e_{k,1} = \frac{1}{\lambda_k} \; ; \; e_{k,i} = \frac{1}{\lambda_k - e_{k,i-1}} \qquad \text{for } i = 2,3,\ldots,N. \tag{44}$$

$$q_{k,1} = \psi^s_{k,1}\, e_{k,1} \; ; \; q_{k,i} = (\psi^s_{k,i} - q_{k,i-1})\, e_i \quad \text{for } i = 2,3,\ldots,N. \tag{45}$$

$$U^s_{k,N} = q_{k,N}; \; U^s_{k,i} = q_{k,i} - e_{k,i}\, U^s_{k,i+1} \quad \text{for } i = N-1,N-2,\ldots,1. \tag{46}$$

This method uses N words of extra storage, N divisions, 2N multiplications, and 3N additions. On ILLIAC IV a division requires as much time as six multiplications. Thus the calculations of the e_i's takes most of the time required for Thomas' method. Since the e_i's are independent of U and Ψ they could be calculated in preconditioning and used repeatedly by Thomas' method. Since each row has a different λ_k we need a different set of $e_{k,i}$'s for each row of U. Thus all the $e_{k,i}$'s would require half as much storage as U if odd/even reduction is applied. Thomas' method can be applied to an arbitrary M. This not only allows the program to be more general but in some cases can save storage in placing U and Ψ in PEM. If a problem with Dirichlet or Neumann boundary conditions required the accuracy obtained by 64 by 64 meshes, applying cyclic reduction would require the meshes to be 65 by 65 while Thomas' method would accept 65 by 64 meshes. In straight storage a 65 by 65 mesh requires 130 lines of PEM while a 65 by 64 mesh requires 65 lines of PEM. If the boundary conditions are periodic, cyclic reduction would require a 64 by 64 mesh, and would require only 64 lines of PEM. The extra storage required to store the $e_{k,i}$'s for Thomas' method is too great for some memory bound problems. By changing (46) to

$$\begin{aligned} &U^s_{k,N} = q_N; \; U^s_{k,N-1} = \psi^s_{k,N} - \lambda_k\, U^s_{k,N}; \\ &U^s_{k,i} = \psi^s_{k,i+1} - \lambda_k\, U_{k,i+1} - U_{k,i+2} \quad \text{for } i = N-2,\ N-3,\ \ldots,\ 1. \end{aligned} \tag{47}$$

we eliminate the need for extra storage. We will refer to this method as modified Thomas' method. It takes N divisions, 2N multiplications, and 4N additions.

Table 8 summarizes the storage and computational time of cyclic reduction, Thomas' method and modified Thomas method where the divisions are done in preconditioning for Thomas' method and cyclic reduction and the divisions are not done in preconditioning for the modified Thomas' method. The computational time is calculated by multiplying 7, 9 and 56 clocks by the number of additions, multiplications and divisions respectively and is for 64 rows of U. The storage is in words not lines of PEM.

	Mesh Size Restrictions	Extra Storage Requirements		Time per 64 Rows in Clocks
Thomas' Method with odd/even reduction	An odd number of columns	Thomas' Method	NM/2 words	39N/2
		Mod. Thomas' Method	No words	102N/2
Thomas' Method without odd/even reduction	No Restrictions	Thomas' Method	NM words	39N
		Mod. Thomas' Method	No words	102N
Cyclic Reduction	$N = 2^I + 1$	$M\log_2 N$ words		$46N - 14\log_2 N$

Table 8. Comparison of Three Methods to Solve Tridiagonal Matrix Equations for Dirichlet's or Neumann's boundary conditions.

Odd/even reduction is most helpful when $\left\lceil \frac{k}{64} \right\rceil = 2 \left\lceil \frac{k}{128} \right\rceil$ where k is the number of interior columns (see Section 2). The time estimates of Table 8 for Thomas' method with odd/even reduction are only half the corresponding values for Thomas' method without odd/even reduction. This is because

odd/even reduction decreases the size of the tridiagonal system to be solved by a factor of 2.

For cyclic reduction the amount of extra storage in Table 8 is misleading. If $I \geq 6$ then 64M words of extra storage are required per mesh in PEM. If both U and Ψ are in PEM,then 128M words are required. This is because cyclic reduction requires $2^I + 1$ columns for Dirichlet's or Neumann's boundary conditions while a similar error bound could be obtained using Thomas' method and 2^I points.

For cyclic reduction on Neumann's boundary conditions, if $I \geq 6$ then an extra Fourier analysis and synthesis must be performed to solve the last column. This time should be divided by the number of rows and added to the time per row for cyclic reduction on Neumann's boundary conditions.

If odd/even reduction is used then the odd columns of U are calculated by solving tridiagonal systems. These systems are already restricted in size by the FFT performed on the even columns. One cyclic reduction program can be written to solve for U on the odd columns for all three boundary conditions. Thus cyclic reduction will be used to solve for U on the odd columns when odd/even reduction is employed.

3.4 Fourier Analysis and Synthesis

Hockney wrote one routine which performs Fourier analysis or synthesis for any of the boundary conditions [Hockney, Page 201]. His routine assumes that there is plenty of temporary storage available. Although this is true on most large serial machines ILLIAC IV has a relatively small amount of storage compared with its speed. Thus a method which is as fast as Hockney's Fourier routine but does not require extra storage would be

preferred. Cooley, the author of FFT, presents such a method in [Cooley, Page 320-323]. Cooley shows that it is possible to use the complex fast Fourier transform to obtain the sine series (required for Dirichlet boundary conditions), cosine series (required for Neumann boundary conditions) and real series (required for periodic boundary conditions). Both Hockney's and Cooley's methods place restrictions on the interior mesh sizes: $2^I - 1$ for Dirichlet, 2^I for periodic and $2^I + 1$ for Neumann.

Cooley's algorithm for Dirichlet boundary conditions goes as follows: let $M = 2^I$, $Y(j)$ for $j = 1, \ldots, M-1$ be the coefficients of the Fourier sine series, $b(k) = \sum_{j=1}^{M-1} Y(j) \sin \frac{\pi jk}{M}$.

$$Y(0) = Y(M) = 0, \text{ and } Y(j) = -Y(2M-j) = -Y(-j) \text{ for } j = 0, \ldots, M.$$

Define

$$X(j) = -[Y(2j+1) - Y(2j-1)] + Y(2j)i \text{ for } j = 0, \ldots, M/2.$$

Thus

$$X(0) = -[Y(1) + Y(-1)] + Y(0) = -2Y(1) + 0i$$

and

$$X(M/2) = -[Y(M+1) - Y(M-1)] + Y(M)i = 2Y(M-1) + 0i.$$

For $j = 1, \ldots, M/2-1$,

$$X(M/2+j) = -Y(M+2j+1) + Y(M+2j-1) + Y(M+2j)i = Y(M-2j-1) - Y(M-2j+1) - Y(M-2j)i.$$

Let $\tilde{X}(j) = C(j)$ and calculate $A_1(j)$ and $A_2(j)$ for $j = 0, 1, \ldots, M/4$ using

$$A_1(j) = C(j) + C(M/2+j) \text{ and } A_2(j) = [C(j) - C(M/2+j)]W_M^j$$

where

$$W_M^j = \text{COS } \frac{2\pi j}{M} + i \text{ SIN } \frac{2\pi j}{M}.$$

Then calculate $A(j)$ and $A(M/4+j)$ for $j = 0, 1, \ldots, M/4-1$

where

$$A(j) = A_1(j) + i\,A_2(j),\quad \tilde{A}(M/4+j) = A_1(j) - iA_2(j).$$

Thus, for $j = 0, 1, \ldots, M/4$,

$$\begin{aligned} A(j) = {} & -Y(2j+1) + Y(2j-1) + Y(M-2j-1) - Y(M-2j+1) + \text{SIN}\,\tfrac{2\pi j}{M}\,[Y(2j+1) \\ & - Y(2j-1) + Y(M-2j-1) - Y(M-2j+1)] + \text{COS}\,\tfrac{2\pi j}{M}\,[Y(2j) + Y(M-2j)] \\ & + i\Big\{Y(M-2j) - Y(2j) + \text{SIN}\,\tfrac{2\pi j}{M}[Y(2j) + Y(M-2j)] + \\ & \text{COS}\,\tfrac{2\pi j}{M}\,[Y(2j-1) - Y(2j+1) - Y(M-2j-1) + Y(M-2j+1)]\Big\} \end{aligned} \tag{48.1}$$

and for $j + 1, \ldots, M/4$

$$\begin{aligned} A(M/2-j) = {} & Y(2j-1) - Y(2j+1) - Y(M-2j+1) + Y(M-2j-1) - \\ & \text{SIN}\,\tfrac{2\pi j}{M}\,[Y(2j+1) - Y(2j-1) + Y(M-2j-1) - Y(M-2j+1)] - \\ & \text{COS}\,\tfrac{2\pi j}{M}\,[Y(2j) + Y(M-2j)] - i\Big\{Y(M-2j) - Y(2j) - \\ & \text{SIN}\,\tfrac{2\pi j}{M}\,[Y(2j) + Y(M-2j)] - \text{COS}\,\tfrac{2\pi j}{M}\,[Y(2j-1) - \\ & Y(2j+1) - Y(M-2j-1) + Y(M-2j+1)]\Big\}. \end{aligned} \tag{48.2}$$

Now the complex fast Fourier transform is applied to A giving

$$X(j) = \sum_{k=0}^{N/2-1} A(k)W_N^{jk},\ \text{for } j = 0, 1, \ldots, M/2-1.$$

For $j = 1, 3, 5, \ldots, M-3$ let

$b(j) = X\left(\frac{j-1}{2}\right)$ imaginary and

$b(j+1) = X\left(\frac{j+1}{2}\right)$ real; $b(m-1) = X\left(\frac{M}{2}-1\right)$ imaginary.

Finally $b(j)$ for $j - 1, \ldots, M-1$ is calculated

$$4b(j) = [b(j) - b(M-j)] - [b(j)+b(M-j)]/[2\text{Sin}\,\tfrac{\pi j}{M}]. \tag{49}$$

Now we have $4b(j) = 4\sum_{k=1}^{M-1} Y(k)\ \text{Sin}\,\frac{\pi jk}{M}$.

Fourier analysis calculates $\frac{2}{N}\,b(j)$ and Fourier synthesis calculates $b(j)$.

Since we are dealing with linear equations we can multiply ρ in (1) and

INITIAL CONDITIONS

REGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS
$X	PI	$C0	I	$D0	J	$D3	KP
$R	P[$X](2)	$C2	I+ILJ	$D1	L	$D4	KM
$S	P[$X](3)	$C3	J-I+ILJ	$D2	K	$D5	ILJ

LOOP	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS	TOTAL TIME FOR LOOP
1	P[$X](3):=$R-$S	PEM Store, Addition	14	
1	$C3:=$C3-$D2	CU Addition overlapped	0	
1	$B:=P[$X](2)	PE Fetch overlapped	0	
1	$A:=$S;$S:=$B	Two PE Register Transfers	2	
1	P[$X](2):=$R+$A	Addition, PE Store	14	
1	$C2:=$C2+$D2	CU Addition overlapped	0	
1	$R:=P[$X](2)	PE Fetch overlapped	0	$30(\frac{M-1}{2})\lceil\frac{N}{64}\rceil$
	P[$X](2):=$R+$R	Addition, PE Store	14	
	$C2:=$D0+$D5-$D2	CU Addition overlapped	0	
	$S:=P[$X-(2)	PE Fetch overlapped	0	
	$C3:=$D5	CU Register Transfer overlapped	0	
	P[$X](3):,-$S-$S	Addition, PE Store	14	
	$C3:=$C3+$D2	CU Addition overlapped	0	
	$R:=P[$X](3) %ODD3	PE Fetch overlapped	0	
	P[$X](3):=-$R-$R	Addition, PE Store	14	
	$C0:=$D3	CU register transfer overlapped	0	$42\lceil\frac{N}{64}\rceil$
2	$C3:=$C3+$D3;$C2:=$C3-$D2	CU Addition overlapped	0	
2	$R:=P[$X](3)-$R	PE Fetch overlapped, Addition	7	
2	$D6:=GRABONE(DS[GL],N2-IL)	CU Load	10	
2	$D7:=GRABONE(DS[GL],IL)	CU Load	10	
2	$S:=$D6*$R	Multiplication	9	
2	$A:=P[$X](2)*$D7	PE Fetch overlapped, Multiplication		
2	$S:=$A-$S %ODD2	Addition	7	
2	$R:=$R*$D7	Multiplication	9	
2	$A:=P[$X](2)*$D6	PE Fetch overlapped	9	
2	$R:=$A+$R %ODD1	Addition	7	
2	$C3:=$D0+$D4-$C0	CU Addition overlapped	0	
2	$C0:=$C0+$D3;$C1:,$C3-$D3	CU Addition overlapped	0	
2	P[$X](2):=P[$X](3)-P[$X](1)	Two PE Fetches, One overlapped Addition, PE Store	21	
2	$C2:=$C2+$D2;$C1:=$C1+$D2	CU Addition overlapped	0	
2	ODD3:=P[$X](2)	PE Fetch overlapped, PE Store	7	
2	P[$X](2):=$S-P[$X](1)	PE Fetch, Addition, PE Store	21	
2	P[$X](1):=$S+P[$X](1)	PE Fetch overlapped, Addition, PE Store	14	
2	$C2:=$C2-$D2	CU Addition overlapped	0	
2	$S:=P[$X](2)	PE fetch overlapped	0	
2	P[$X](1):=$S-$R	Addition, PE Store	14	
2	$B:=$R	PE Register Transfer overlapped	0	
2	$R:=ODD3	PE Fetch overlapped	0	
2	P[$X](2):=$S+$B	Addition, PE Store	14	$159(\frac{M-1}{4}-1)\lceil\frac{N}{64}\rceil$
	$C2:=$D4+$D1;$C3:=$D4+$D5	CU Addition overlapped	0	
	$R:=P[$X](2)	PE Fetch overlapped	0	
	P[$X](2):=P[$X](3)	PE Fetch, PE Store	14	
	P[$X](3):=$R+$R	Addition, PE Store	14	$28\lceil\frac{N}{64}\rceil$
		SUBTOTAL		$(57M-155)\lceil\frac{N}{64}\rceil$

Table 9. Preparation of the Data for the Fast Fourier Transform*

*The logic for the code to keep current values in the ADB's defined in the initial conditions is supplied in GLYPNIR, Appendix B.

INITIAL CONDITIONS							
REGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS
$X	PI	$D2	K	$D5	ILJ	$D8	I2
$D0	J	$D3	KP	$D6	IP	$D9	CJ
$D1	L	$D4	KM	$D7	J1	$D10	IT

LOOP	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS	TOTAL TIME FOR LOOP
	$C2;=$D5	CU Register transfer overlapped	0	0
3	$C3:=$C2+$D1	CU Addition overlapped	0	
3	$S:=P[$X](2)	PE Fetch overlapped	0	
3	P[$X](2):=$S+P[$X](3)	PE Fetch, Addition, PE Store	21	
3	P[$X](3):=$S-P[$X](3)	PE Fetch overlapped, Addition, PE Store	14	
3	$C2:=$C2+$D2	CU Addition overlapped	0	$35(\frac{M+1}{2})\lceil\frac{N}{64}\rceil$
	$C0:=$D3	CU Register Transfer overlapped	0	
	$C2;=GRABONE(INDEX[GL],IL);	CU Load	10	$10(\frac{M-5}{2})$
4	$R:=P(0)	PE Fetch	7	
4	P(0):=P(2)	PE Fetch, PE Store	14	
4	P(2):=$R	PE Store	7	
4	$C0;=$C0+1;$C2;=$C2+1	CU Addition overlapped	0	$28(\frac{M-5}{2})\lceil\frac{N}{64}\rceil$
	$D11:=GRABONE(DS[GL],N2-N11) *$D8	CU Load; Multiplication	19	
	$D12:=GRABONE(DS[GL],N11)	CU Load	10	29(M-1)
5	$C2:=$D4+$D7+$D5	CU Addition overlapped	0	
5	$C3:=$C2+$D9+$D9;$C1:= $C3+$C2	CU Addition overlapped	0	
5	$R:=P[$X](3)*$D11	PE Fetch overlapped, Multiplication	9	
5	$S:=P[$X](1)	PE Fetch overlapped	0	
5	$B:=$S*$D12	Multiplication	9	
5	$R:=$R-$B %ODD1	Addition	7	
5	$S:=$S*$D11	Multiplication	9	
5	$A:=P[$X](3)*$D12	PE Fetch overlapped, Multiplication	9	
5	$S:=$A+$S %ODD2	Addition	7	
5	P[$X](3):=P[$X](2)-$R	PE Fetch overlapped, Addition, PE Store	14	
5	P[$X](2):=P[$X](2)+$R	PE Fetch overlapped, Addition, PE Store	14	
5	$C2:=$C2+$D2	CU Addition overlapped	0	
5	P[$X](1):=P[$X](2)-$S	PE Fetch overlapped, Addition, PE Store	14	
5	P[$X](2):=P[$X](2)+$S	PE Fetch overlapped, Addition, PE Store	14	$106(\text{Log}_2(M)-2)(\frac{M+1}{8})\lceil\frac{N}{64}\rceil$
		SUBTOTAL		$(\frac{53}{4}(M+1)\text{Log}_2(M)+5M-79)\lceil\frac{N}{64}\rceil+34M-54$

Table 10. The Complex Fast Fourier Transform*

* The logic for the code to keep current values in the ADB's defined in the initial conditions is supplied in GLYPNIR, Appendix B.

INITIAL CONDITIONS							
REGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS
$X	PI	$D0	J	$D1	I	$D2	ILJ

LOOP	ASSIGNMENT STATEMENT	OPERATIONS AND OVERLAP	TIME IN CLOCKS	TOTAL TIME FOR LOOP
	$D3:=GRABONE(IS[GL],I1)	CU Load	10	$10(\frac{M+1}{2})$
6	$C1:=$D2+$D1;$C2:= $D0-$D1+$D2	CU Addition overlapped	0	
6	$R:=P[$X](1)	PE Fetch overlapped	0	
6	$B:=P[$X](2)	PE Fetch	7	
6	$R:=$R=$B	Addition	7	
6	$A:=P[$X](1)	PE Fetch overlapped	0	
6	$C3:=$D3	CU Register transfer overlapped	0	
6	$A:=$A+$B	Addition	7	
6	$S:=$A*$C3	Multiplication	9	
6	P[$X](1):=$R+$S	Addition, PE Store	14	
6	P[$X](2):=$S-$R	Addition, PE Store	14	$58(\frac{M+1}{2})\lceil\frac{N}{64}\rceil$
		SUBTOTAL	$29M\lceil\frac{N}{64}\rceil+29\lceil\frac{N}{64}\rceil+5M+5$	

Table 11. Final Step in Cooley's Fourier Analysis and Synthesis*

*The logic for the code to keep current values in the ADB's defined in the initial conditions is supplied in GLYPNIR, Appendix B.

INITIAL CONDITIONS							
EGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS	REGISTER	CONTENTS
$X	PI	$D1	IL	$D3	L3	$D5	L4
$DO	CI	$D2	L2	$D4	K	$D6	KM1

OOP	ASSIGNMENT STATEMENTS	OPERATIONS AND OVERLAP	TIME IN CLOCKS	TOTAL TIME FOR LOOP
7	$C2:=$D1;$C3:=$DO	CU Register Transfer overlapped	0	
7	P(2):=RTR($C3,,P(2));	PE Fetch, Route, PE Store	20	$20(M-1)\left\lceil\frac{N}{64}\right\rceil$
	$C1:=0;$C3:+$DO	CU Register clear overlapped	0	
	$S:=1.0/A(3)	PE Fetch, division	63	
	$B:=P[$X]*$S	PE Fetch overlapped, Multiplication	9	$72\left\lceil\frac{M}{64}\right\rceil$
8	$C1:=$C1+1	CU Addition overlapped		
8	$A:=RTR($C1,,A(3))	PE Fetch, overlapped, Route	6	
8	$S:=RTR(1,,$S)	Route	3	
8	$X;=RTR(1,,$X)	Route	3	
8	$R:=RTR(1,,$B)	Route	3	
8	$A:=$A-$S	Addition	7	
8	$S:=1.0/$A %E	Division	56	
8	$A:=P[$X]-$R	PE Fetch overlapped, Addition	7	
8	$B;=$A*$S %Q	Multiplication	9	$94(N-1)\left\lceil\frac{M}{64}\right\rceil$
	$A:=RTR($C1,,A(3))	PE Fetch overlapped, Route	6	
	$R:+$B	PE Register transfer overlapped	0	
	$A:=$A*$B	Multiplication	9	
	$C2:=$D6	CU Register transfer overlapped	0	
	$S;=P[$X](2)-$A	PE Fetch overlapped, Addition	7	
	P[$X](2):=$R	PE Store overlapped	0	$22\left\lceil\frac{M}{64}\right\rceil$
9	$C2:=$D2+$D3;$C1:=$C1-1	CU Addition overlapped	0	
9	$B:=RTL(1,,$R) %Q	Route	3	
9	$S:=RTL(1,,$S) %E	Route	3	
9	$X:=RTL(1,,$X) %PI	Route	3	
9	$A;=RTR($C1,,A(3))	PE Fetch, partially overlapped Route	10	
9	$R:=$B %Q	PE Register transfer overlapped	0	
9	$A:=$A*$S	Multiplication	9	
9	$A;=P[$X](2)-$A	PE Fetch overlapped, Addition	7	
9	$A;=$A=$R %T	Addition	7	
9	P[$X](2):=$S	PE Store overlapped	0	
9	$B:=$S %Q	PE Register transfer	1	
9	$S:=$A %E	PE Register transfer	1	$46(M-1)\left\lceil\frac{M}{64}\right\rceil$
)	%C2:=$D1;$C3:=$DO	CU Register transfer overlapped	0	
)	P(2):=RTL($C3,,P(2))	PE Fetch, Route, PE Store	20	$20(M-1)\left\lceil\frac{N}{64}\right\rceil$
		SUBTOTAL	$14ON\left\lceil\frac{M}{64}\right\rceil-46\left\lceil\frac{M}{64}\right\rceil$	$+40(M-1)\left\lceil\frac{N}{64}\right\rceil$

Table 12. CRED*

*The logic for the code to keep current values in the ADB's defined in the initial conditions is supplied in GLYPNIR, Appendix B.

the boundary conditions by $\frac{1}{8M}$. Then apply (48.1), (48.2), the complex fast Fourier transform and (49) before and after CRED. This algorithm, MFACR, is a modified FACR which solves (39) for Dirichlet's boundary conditions without using odd/even reduction.

3.5 Implementation of MFACR

We will discuss MFACR in explaining how it can be programmed in ASK and how much time such a routine would take. Appendix F contains GLYPNIR code of the algorithm. Appendix A contains information on timing methodology and explains the notation used in Tables 9 through 12. Both Ψ and U are M+2 by N+2 mesh. Ψ and U are stored in straight storage in C and P, respectively. MFACR consists of five steps:

1. Set up storage area P
2. Fourier analysis uses Ψ to calculate Ψ^S
3. CRED uses Ψ^S to calculate U^S
4. Fourier synthesis uses U^S to calculate U
5. Restore boundary conditions and C.

Step one places a linear combination of the boundary conditions and Ψ in the interior of the storage area P. In the process the first row of U along with the second and N+1st row of Ψ are lost. Thus they are saved in temporary storage. This section of the algorithm takes $(23M+102)\left\lceil \frac{N}{64} \right\rceil + 13M$ clocks in ASK. Section 1 of Appendix F contains the GLYPNIR code.

Fourier synthesis and analysis are performed on up to 64 columns simultaneously described above. They are presented in three parts. First equations (48.1) and (48.2) prepare the data for the complex

fast Fourier transform. Table 9* contains suggestions on how these equations could be programmed in ASK and supplies $(57M-155)\left\lceil\frac{N}{64}\right\rceil$ clocks as a time estimate for this step. Table 10 contains suggestions on how the complex fast Fourier transform could be programmed in ASK. A time estimate of $(\frac{53}{4}(M+1)\,\mathrm{Log}_2(M+1) + 5M-79)\left\lceil\frac{N}{64}\right\rceil + 34M - 54$ is given for this step. Finally the implementation of equation (49) in ASK is discussed in Table 11 where a time estimate of $29(M+1)\left\lceil\frac{N}{64}\right\rceil + 5M + 5$ is presented. The total time estimate for Cooley's algorithm is $\{\frac{53}{4}(M+1)\,\mathrm{Log}_2(M+1) + 91M - 205\}\left\lceil\frac{N}{64}\right\rceil + 39M - 49$ clocks. The GLYPNIR code for Cooley's method is found in section 2 of Appendix F.

MFACR uses the modified Thomas' method, equations (44), (45), and (47) to perform CRED on up to 64 rows simultaneously. Table 12 suggests how this method could be programmed in ASK and estimates how long that code would take, $40(M-1)\left\lceil\frac{N}{64}\right\rceil + (140N-46)\left\lceil\frac{M}{64}\right\rceil$ clocks. Section 3 of Appendix F contains the GLYPNIR code for CRED.

The final step of MFACR consists of restoring the boundary conditions and the portions Ψ which have been altered. Section 4 of Appendix F contains the GLYPNIR code for this step. If programmed in ASK this step would take approximately $42\left\lceil\frac{N}{64}\right\rceil$ clocks.

The complete MFACR requires $\{53/2(M-1)\,\mathrm{Log}_2(M+1) + 245M-306\}\left\lceil\frac{N}{64}\right\rceil + (140N-46)\left\lceil\frac{M}{64}\right\rceil + 91M - 98$ clocks. The timing algorithm is taken from Appendix A.

---

* The loop numbers in Tables 9 to 12 correspond to the loop number in the GLYPNIR code in Appendix F. They are supplied to enable the reader to compare loops in the GLYPNIR code with the corresponding loops in the ASK code.

3.6 Implementation of FACR

FACR, one level of odd/even reduction would require $(53/2(M-1)\log(M+1) + 405M - 466)\left\lceil\frac{N}{128}\right\rceil + (80M + 104)\left\lceil\frac{N}{64}\right\rceil + (70N - 23)\left\lceil\frac{M}{64}\right\rceil + 81M - 84$ clocks. This figure assumes the odd columns are solved by Thomas' method with the e's calculated and stored in preconditioning. Thus the odd columns would take $(84M - 46)\left\lceil\frac{N}{128}\right\rceil$ clocks. Setting up the even columns would take $79M\left\lceil\frac{N}{128}\right\rceil$. Preparing the odd rows for solution would take $55M\left\lceil\frac{N}{128}\right\rceil$ clocks. CRED would take $(70N-23)\left\lceil\frac{M}{64}\right\rceil$ clocks. Fourier analysis and synthesis would take $(53/2(M-1)\log_2(M+1) + 192M - 420)\left\lceil\frac{N}{128}\right\rceil + 68M - 48$. The extra storage manipulation would take $17M\left\lceil\frac{N}{64}\right\rceil$ clocks. The rest of the algorithm is identical to MFACR.

3.7 Summary of the Results on Direct Methods

In programming an algorithm on ILLIAC IV, an attempt is made to adapt the restrictions of the algorithm to maximize storage and machine efficiency. For efficient storage on ILLIAC IV, rows stored across PEM should contain a multiple of 64 words. To maximize machine efficiency the number of values being computed simultaneously equals 64. Hockney's method performs a Fourier analysis and synthesis on each column and performs CRED on each row. Thus to maximize machine efficiency on ILLIAC IV the number of interior rows and the number of interior columns must be multiples of 64. Fourier analysis and synthesis using Cooley's method requires $2^I - 1$ points per column for Dirichlet's and Neumann's boundary conditions. If $I \geq 6$ then storing the columns across PEM would waste 63 words per column. Thus we will store each column in one PE. Now we have columns restricted to 2^I+1 (2^I) for

Dirichlet's or Neumann's (periodic) boundary conditions. Rows are restricted to multiples of 64 to maximize storage efficiency. For periodic boundary conditions cyclic reduction saves time and temporary storage over Thomas' method [Hockney, Page 150], and restricts the mesh to 2^I columns. This restriction on mesh size still allows maximum storage and machine efficiency. Cyclic reduction restricts the mesh to $2^I + 1$ for Dirichlet's and Neumann's boundary conditions This restriction reduces storage efficiency for both boundary conditions and it reduces machine efficiency for Neumann's boundary conditions. Thomas' method and the modified Thomas' method have no restrictions on the number of columns. Thomas' method is faster than the modified Thomas' method but requires temporary storage. Refer to Table 8 for a more detailed comparison of the methods available for use by CRED.

Hockney's FACR program handles the 9 different combinations of boundary conditions and performs one level of odd/even reduction in all cases. To maximize efficiency of such a program on ILLIAC IV different algorithms would be required for different mesh sizes and combinations of boundary conditions.

As stated earlier, odd/even reduction saves computer time on ILLIAC IV for certain mesh sizes but not for others. Thus FACR on ILLIAC IV should have two algorithms, one which uses odd/even reduction and one that does not. Furthermore, the choice of a method to solve CRED depends on the storage requirements of the individual problem. It is the author's opinion that modified Thomas' Method is the best compromise between storage and speed for Neumann's and Dirichlet's boundary conditions, and cyclic reduction is the best for periodic boundary conditions.

If Fourier analysis and synthesis is applied to columns which have Neumann's boundary conditions then there must be $2^I + 1$ interior rows to be evaluated by CRED. This means CRED must be executed $\left\lceil \frac{2^I+1}{64} \right\rceil$ times. Note $\left\lceil \frac{2^I+1}{64} \right\rceil - \frac{2^I}{64} + 1$ if $I \geq 6$ and thus for one execution of CRED, ILLIAC IV is working at $1/64^{th}$ of its capacity. The user can avoid this machine inefficiency by setting up the program so that Fourier analysis and synthesis is performed in the direction which has non-Neumann boundary conditions.

On a M by N mesh the Fourier part of FACR requires the order of $NM\log_2 M$ clocks when it is applied to the N columns in the mesh. CRED requires the order of NM clocks. By letting M be the smaller of the dimensions the user saves computer time.

4. USE OF ILLIAC DISK FOR NON-CORE CONTAINED MESHES

When the problem being run becomes too large to contain in core, the I/O time becomes an important consideration. First the I/O system for ILLIAC IV will be discussed and then its efficiency for a specific problem will be examined.

4.1 ILLIAC IV I/O System

The ILLIAC IV Disk is a 15,600 K word memory. This memory is divided into 52 bands. Each band is divided into 300 pages, each containing 16 lines of PEM (1 K words). Memory transfers between the ILLIAC IV Disk and PEM are restricted to pages. A data request can read or write up to 128 consecutive pages on one band. If the data is not consecutive or if it is spread over more than one band a separate request is needed for each string of consecutive pages. The time required for the disk to prepare to perform a data request is equivalent to the time to transfer two pages of data between the Disk and PEM. Thus if a data request reads or writes on page i and band j, the next data request should skip at least two pages and start at page i + 3 of band k, where j need not equal k. If the second data request wanted page i + 1, or i + 2, the request would have to wait for a revolution of the disk before it could be executed. The transfer rate between PEM and Disk is 133 usec per page and the disk revolves once every 40 msec. One revolution of the disk is equivalent to 640,000 ILLIAC IV clocks.

4.2 I/O for the Bernard-Rayleigh convection problem

Now let us examine the Bernard-Rayleigh convection problem where the following equations are used to solve for temperature, T, pressure, P, and velocity components u and w along with the X and Z axes:

$$\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + w \frac{\partial T}{\partial z} = \frac{1}{P_r} \nabla^2 T \tag{50}$$

$$\frac{\partial \eta}{\partial t} + u \frac{\partial \eta}{\partial x} + w \frac{\partial \eta}{\partial z} = \frac{R_a}{P_r} \frac{\partial T}{\partial x} \nabla^2 \eta. \tag{51}$$

$$\eta = \frac{\partial w}{\partial x} - \frac{\partial u}{\partial z} = \nabla^2 \psi \tag{52}$$

where t is time, R_a is the Rayleigh number, P_r is the Prandtl number, and ψ is the stream function defined by

$$u = - \frac{\partial \psi}{\partial z}, \quad w = \frac{\partial \psi}{\partial x}$$

The finite difference schemes employed to solve these equations require five meshes: $T^{(\tau)}$, $T^{(\tau-1)}$, $\eta^{(\tau)}$, $\eta^{(\tau-1)}$ and $\psi^{(\tau)}$.

Each time step of the convection process has three main parts; calculate $\eta^{(\tau+1)}$, calculate $T^{(\tau+1)}$ and calculate $\Psi^{(\tau+1)}$. $\eta^{(\tau+1)}_{i,j}$ is calculated using $\eta^{(\tau)}_{i\pm1,j}$, $\eta^{(\tau)}_{i,j\pm1}$, $\eta^{(\tau-1)}_{i,j}$, $T^{(\tau)}_{i\pm1,j}$, and $\Psi^{(\tau)}_{i\pm1,j\pm1}$. $T^{(\tau+1)}_{i,j}$ is calculated using $T^{(\tau)}_{i\pm1,j}$, $T^{(\tau)}_{i,j\pm1}$, $T^{(\tau-1)}_{i,j}$ $\Psi^{(\tau)}_{i\pm1,j\pm1}$, $\Psi^{(\tau)}_{i\pm1,j}$ and $\Psi^{(\tau)}_{i,j\pm1}$. Using all of $\eta^{(\tau+1)}$, $\Psi^{(\tau+1)}_{i,j}$ is calculated.

The process will be divided into two parts. First the prognostic equations (50) and (51) are solved to obtain $\eta^{(\tau+1)}$ and $T^{(\tau+1)}$. Then Poisson's equation, (52), is solved for $\Psi^{(\tau+1)}$.

The prognostic equations require five meshes while Poisson's equation requires two meshes if solved by an iterative method and one mesh if solved by a direct method. Thus non-core contained problems can be divided into two types. In the first type, the meshes required for solving Poisson's equation are core contained but some of the meshes required for solving the prognostic equations are not contained in core.

In the second case, the meshes are so large that none of the meshes can be core contained.

In [Ogura, et al] the first type of problem was studied in which SOR was used to solve Poisson's equation. If a direct method had been used in that study, the maximum mesh size could be doubled.

In this study the second type of I/O problem will be examined where MFACR is used to solve Poisson's equation. For an example we will use 512 by 512 meshes. $P_{i,j}$ where $1 \le i \le 64$ and $0 \le j \le 3$ is a page of a mesh containing all points $a_{k,\ell}$ where $8_i - 8 \le k < 8_i$ and $128\ j \le \ell < 128\ j+128$.

If the prognostic equations are solved separately using ASK code they are found to be about 50% I/O bound. To decrease the I/O bound the Fourier analysis of MFACR will be performed along with the calculation of the prognostic equations.

This will be followed by CRED, with the final step being Fourier synthesis.

The data will be stored in blocks 22 pages long. Block number I, B_I contains $P_{I,j}$ where $0 \le j \le 3$ of $\eta^{(\tau)}$, $T^{(\tau)}$, $\eta^{(\tau-1)}$, $T^{(\tau-1)}$, and $\Psi^{(\tau)}$ for time step τ. Also included in B_I are the two pages required for the read or write interpret. Thirteen blocks can be placed on one band with 14 pages remaining. The data will be spread across the 11 bands of disk with 22 pages separating each block to leave room for the information from PEM to be written. Seven blocks of data and six blocks for writing are found on bands 0, 2, 4, 6, and 8. Six blocks of data and seven blocks for writing are found on bands 1, 3, 5, and 7. Band 9 contains 5 blocks of data and seven blocks for writing. Band 10 contains the last block for writing.

The blocks are skewed to enable efficient access of rows and columns. The first block on band I starts at page -2I mod 300.

The prognostic equations and the Fourier analysis are performed while the disk revolves ten times and 110 pages are passed on the eleventh revolution. Figure 6 shows what data will be printed on the first 44 pages of each band.

Only one column of pages can fit in PEM at one time. Since CRED requires an entire column before the calculations can be performed, a column is read into PEM, the calculations are performed and the column is printed on disk. The data has been skewed in such a way as to enable an entire column to be read or written in one revolution, see Figure 6. There are four columns and the calculation of each takes 1/4 of a revolution. The total I/O time for CRED is nine revolutions.

The final step, Fourier Synthesis, is performed in four revolutions. Figure 6 shows that this can be accomplished on the first 44 pages by calculating rows 5, 31 and 57 on the first revolution, rows 18 and 44 on the second revolution, rows 12, 38, 64 on the third revolution and rows 25, 51 on the last revolution. To enable $\Psi^{(\tau+1)}$ to overwrite $\Psi^{(\tau+1/3)}$ Row I of $\Psi^{(\tau+2/3)}$ is written 3 data blocks before Row I of $\Psi^{(\tau+1/3)}$ during CRED.*

The entire time step required 23 revolutions and 110 pages on the 24th revolution. This takes 932 milliseconds. The calculation could be performed in 333 milliseconds. Thus the process is 64% I/O bound.

A large portion of this I/O bound is created by the solution of Poisson's equation. The last two steps of MFACR required 13 revolutions of the disk while the calculations for those steps required the amount of time for $2\frac{1}{2}$ revolutions.

* $\Psi^{(\tau+1/3)}$ contains the harmonics calculated by Fourier analysis.
$\Psi^{(\tau+2/3)}$ contains the harmonics calculated by CRED.

To minimize total computer time a problem takes, a programmer must adapt algorithms to disk maps which give minimum I/O time. With I/O being such a dominant factor in the above problem the use of library subroutines becomes minimal because programs need to be customized to minimize I/O.

Figure 6. A Disk Map of 512 by 512 Meshes

Page	Band 0	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6	Band 7	Band 8	Band 9	Ban 10
0		$T_{5,0}^{(\tau+1)}$		$\eta_{18,0}^{(\tau+1)}$		$T_{31,0}^{(\tau)}$		$\eta_{44,0}^{(\tau)}$		$\Psi_{57,0}^{(\tau+\frac{1}{3})}$	
1		$T_{5,1}^{(\tau+1)}$		$\eta_{18,1}^{(\tau+1)}$		$T_{31,1}^{(\tau)}$		$\eta_{44,1}^{(\tau)}$		$\Psi_{57,1}^{(\tau+\frac{1}{3})}$	
2		$T_{5,2}^{(\tau+1)}$		$\eta_{18,2}^{(\tau+1)}$		$T_{31,2}^{(\tau)}$		$\eta_{44,2}^{(\tau)}$		$\Psi_{57,2}^{(\tau+\frac{1}{3})}$	
3		$T_{5,3}^{(\tau+1)}$		$\eta_{18,3}^{(\tau+1)}$		$T_{31,3}^{(\tau)}$		$\eta_{44,3}^{(\tau)}$		$\Psi_{57,3}^{(\tau+\frac{1}{3})}$	
4		$\eta_{5,0}^{(\tau+1)}$		$T_{18,0}^{(\tau)}$		$\eta_{31,0}^{(\tau)}$		$\Psi_{44,0}^{(\tau+\frac{1}{3})}$			$T_{64}^{(\tau}$
5		$\eta_{5,1}^{(\tau+1)}$		$T_{18,1}^{(\tau)}$		$\eta_{31,1}^{(\tau)}$		$\Psi_{44,1}^{(\tau+\frac{1}{3})}$			$T_{64}^{(\tau}$
6		$\eta_{5,2}^{(\tau+1)}$		$T_{18,2}^{(\tau)}$		$\eta_{31,2}^{(\tau)}$		$\Psi_{44,2}^{(\tau+\frac{1}{3})}$			$T_{64}^{(\tau}$
7		$\eta_{5,3}^{(\tau+1)}$		$T_{18,3}^{(\tau)}$		$\eta_{31,3}^{(\tau)}$		$\Psi_{44,3}^{(\tau+\frac{1}{3})}$			$T_{64}^{(\tau}$
8		$T_{5,0}^{(\tau)}$		$\eta_{18,0}^{(\tau)}$		$\Psi_{31,0}^{(\tau+\frac{1}{3})}$			$T_{51,0}^{(\tau+1)}$		$\eta_{64}^{(\tau}$
9		$T_{5,1}^{(\tau)}$		$\eta_{18,1}^{(\tau)}$		$\Psi_{31,1}^{(\tau+\frac{1}{3})}$			$T_{51,1}^{(\tau+1)}$		$\eta_{641}^{(\tau}$
10		$T_{5,2}^{(\tau)}$		$\eta_{18,2}^{(\tau)}$		$\Psi_{31,2}^{(\tau+\frac{1}{3})}$			$T_{51,2}^{(\tau+1)}$		$\eta_{642}^{(\tau}$
11		$T_{5,3}^{(\tau)}$		$\eta_{18,3}^{(\tau)}$		$\Psi_{31,2}^{(\tau+\frac{1}{3})}$			$T_{51,3}^{(\tau+1)}$		$\eta_{643}^{(\tau}$
12		$\eta_{5,0}^{(\tau)}$		$\Psi_{18,0}^{(\tau+\frac{1}{3})}$			$T_{38,0}^{(\tau+1)}$		$\eta_{51,0}^{(\tau+1)}$		$T_{64}^{(\tau}$
13		$\eta_{5,1}^{(\tau)}$		$\Psi_{18,1}^{(\tau+\frac{1}{3})}$			$T_{38,1}^{(\tau+1)}$		$\eta_{51,1}^{(\tau+1)}$		$T_{641}^{(\tau}$
14		$\eta_{5,2}^{(\tau)}$		$\Psi_{18,2}^{(\tau+\frac{1}{3})}$			$T_{38,2}^{(\tau+1)}$		$\eta_{51,2}^{(\tau+1)}$		$T_{642}^{(\tau}$
15		$\eta_{5,3}^{(\tau)}$		$\Psi_{18,3}^{(\tau+\frac{1}{3})}$			$T_{38,3}^{(\tau+1)}$		$\eta_{51,3}^{(\tau+1)}$		$T_{643}^{(\tau}$
16		$\Psi_{5,0}^{(\tau+\frac{1}{3})}$			$T_{25,0}^{(\tau+1)}$		$\eta_{38,0}^{(\tau+1)}$		$T_{51,0}^{(\tau)}$		$\eta_{64}^{(\tau}$
17		$\Psi_{5,1}^{(\tau+\frac{1}{3})}$			$T_{25,1}^{(\tau+1)}$		$\eta_{38,1}^{(\tau+1)}$		$T_{51,1}^{(\tau)}$		$\eta_{641}^{(\tau}$
18		$\Psi_{5,2}^{(\tau+\frac{1}{3})}$			$T_{25,2}^{(\tau+1)}$		$\eta_{38,2}^{(\tau+1)}$		$T_{51,2}^{(\tau)}$		$\eta_{642}^{(\tau}$
19		$\Psi_{5,3}^{(\tau+\frac{1}{3})}$			$T_{25,3}^{(\tau+1)}$		$\eta_{38,3}^{(\tau+1)}$		$T_{51,3}^{(\tau)}$		$\eta_{643}^{(\tau}$
20			$T_{12,0}^{(\tau+1)}$		$\eta_{25,0}^{(\tau+1)}$		$T_{38,0}^{(\tau)}$		$\eta_{51,0}^{(\tau)}$		$\Psi_{64}^{(\tau}$

21	$T_{12,1}^{(\tau+1)}$		$\eta_{25,1}^{(\tau+1)}$		$T_{38,1}^{(\tau)}$		$\eta_{51,1}^{(\tau)}$		$\psi_{64,1}^{(\tau+\frac{1}{3})}$
22	$T_{12,2}^{(\tau+1)}$		$\eta_{25,2}^{(\tau+1)}$		$T_{38,2}^{(\tau)}$		$\eta_{51,2}^{(\tau)}$		$\psi_{64,2}^{(\tau+\frac{1}{3})}$
23	$T_{12,3}^{(\tau+1)}$		$\eta_{25,3}^{(\tau+1)}$		$T_{38,3}^{(\tau)}$		$\eta_{51,3}^{(\tau)}$		$\psi_{64,3}^{(\tau+\frac{1}{3})}$
24	$\eta_{12,0}^{(\tau+1)}$		$T_{25,0}^{(\tau)}$		$\eta_{38,0}^{(\tau)}$		$\psi_{51,0}^{(\tau+\frac{1}{3})}$		
25	$\eta_{12,1}^{(\tau+1)}$		$T_{25,1}^{(\tau)}$		$\eta_{38,1}^{(\tau)}$		$\psi_{51,1}^{(\tau+\frac{1}{3})}$		
26	$\eta_{12,2}^{(\tau+1)}$		$T_{25,2}^{(\tau)}$		$\eta_{38,2}^{(\tau)}$		$\psi_{51,2}^{(\tau+\frac{1}{3})}$		
27	$\eta_{12,3}^{(\tau+1)}$		$T_{25,3}^{(\tau)}$		$\eta_{38,3}^{(\tau)}$		$\psi_{51,3}^{(\tau+\frac{1}{3})}$		
28	$T_{12,0}^{(\tau)}$		$\eta_{25,0}^{(\tau)}$		$\psi_{38,0}^{(\tau+\frac{1}{3})}$			$T_{58,0}^{(\tau+1)}$	
29	$T_{12,1}^{(\tau)}$		$\eta_{25,1}^{(\tau)}$		$\psi_{38,1}^{(\tau+\frac{1}{3})}$			$T_{58,1}^{(\tau+1)}$	
30	$T_{12,2}^{(\tau)}$		$\eta_{25,2}^{(\tau)}$		$\psi_{38,2}^{(\tau+\frac{1}{3})}$			$T_{58,2}^{(\tau+1)}$	
31	$T_{12,3}^{(\tau)}$		$\eta_{25,3}^{(\tau)}$		$\psi_{38,3}^{(\tau+\frac{1}{3})}$			$T_{58,3}^{(\tau+1)}$	
32	$\eta_{12,0}^{(\tau)}$		$\psi_{25,0}^{(\tau+\frac{1}{3})}$			$T_{45,0}^{(\tau+1)}$		$\eta_{58,0}^{(\tau+1)}$	
33	$\eta_{12,1}^{(\tau)}$		$\psi_{25,1}^{(\tau+\frac{1}{3})}$			$T_{45,1}^{(\tau+1)}$		$\eta_{58,1}^{(\tau+1)}$	
34	$\eta_{12,2}^{(\tau)}$		$\psi_{25,2}^{(\tau+\frac{1}{3})}$			$T_{45,2}^{(\tau+1)}$		$\eta_{58,2}^{(\tau+1)}$	
35	$\eta_{12,3}^{(\tau)}$		$\psi_{25,3}^{(\tau+\frac{1}{3})}$			$T_{45,3}^{(\tau+1)}$		$\eta_{58,3}^{(\tau+1)}$	
36	$\psi_{12,0}^{(\tau+\frac{1}{3})}$			$T_{32,0}^{(\tau+1)}$		$\eta_{45,0}^{(\tau+1)}$		$T_{58,0}^{(\tau)}$	
37	$\psi_{12,1}^{(\tau+\frac{1}{3})}$			$T_{32,1}^{(\tau+1)}$		$\eta_{45,1}^{(\tau+1)}$		$T_{58,1}^{(\tau)}$	
38	$\psi_{12,2}^{(\tau+\frac{1}{3})}$			$T_{32,2}^{(\tau+1)}$		$\eta_{45,2}^{(\tau+1)}$		$T_{58,2}^{(\tau)}$	
39	$\psi_{12,3}^{(\tau+\frac{1}{3})}$			$T_{32,3}^{(\tau+1)}$		$\eta_{45,3}^{(\tau+1)}$		$T_{58,3}^{(\tau)}$	
40		$T_{19,0}^{(\tau+1)}$		$\eta_{32,0}^{(\tau+1)}$		$T_{45,0}^{(\tau)}$		$\eta_{58,0}^{(\tau)}$	
41		$T_{19,1}^{(\tau+1)}$		$\eta_{32,1}^{(\tau+1)}$		$T_{45,1}^{(\tau)}$		$\eta_{58,1}^{(\tau)}$	
42		$T_{19,2}^{(\tau+1)}$		$\eta_{32,2}^{(\tau+1)}$		$T_{45,2}^{(\tau)}$		$\eta_{58,2}^{(\tau)}$	
43		$T_{19,3}^{(\tau+1)}$		$\eta_{32,3}^{(\tau+1)}$		$T_{45,3}^{(\tau)}$		$\eta_{58,3}^{(\tau)}$	

5. CONCLUSIONS

Table 13 compares MFACR and FACR with SOR and ADI for Dirichlet's boundary conditions. The direct methods use modified Thomas' method to perform CRED and Cooley's methods to perform the Fourier part of the algorithm. Unless an excellent initial guess is supplied for the iterative methods, the direct methods are much faster.

For maximum machine efficiency, MFACR requires 64k by 2^I+1 meshes, FACR requires 128k by 2^I+1 meshes, ADI requires 64k by 64I meshes, and SOR requires 64k by j meshes. As seen in section 2, SOR can partition its mesh to obtain maximum machine efficiency. If the mesh for one of the other methods did not meet the row restrictions, the mesh could not be partitioned to meet the restrictions. A possible solution would be to solve a number of meshes at the same time.

MESH SIZE M+2 by N+2	FACR	MFACR	SOR PER ITERATION	ADI PER ITERATION
33 by 63	6.51 or 25300	5.71 or 22400	I or 3900	3.81 or 14900
65 by 63	5.31 or 49500	4.9I or 39100	I or 7900	2.51 or 20100
65 by 127	3.71 or 59500	4.71 or 74000	I or 15900	2.61 40700
129 by 127	3.81 or 122000	4.9I or 156000	I or 32000	2.71 or 87700

Table 13. A Comparison of Methods with Respect to Time*

---

* Two numbers are supplied. The number times I is the number of iterations which could be performed in an equivalent amount of time. The other number is the number of clocks required on ILLIAC IV. The times given for SOR in section 2.3 don't take into account the time required to check for convergence. To calculate the times for SOR in Table 13,20% has been added to the times in section 2.3 to account for convergence checking.

Direct methods require only one mesh in core. The mesh initially contains the interior source points and the boundary conditions. Throughout the algorithm, the mesh is used as temporary storage with the values in the mesh set equal to the solution in the final step. Iterative methods require both the source and the solution mesh at every iteration.

REFERENCES

Birkoff, G. and MacLane, S. A Survey of Modern Algebra. New York: The MacMillan Company, 1962.

Cooley, J. W., et al. "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine, and LaPlace Transform." Journal of Sound and Vibrations, vol. 12, no. 2, June 1970.

Cuthill, E. H. and Varga, R. S. "A Method of Normalized Block Iteration." Journal Assoc. Comput. Mach. 6 (1959): 236-244.

Denenberg, S. A. "An Introduction Description of the ILLIAC IV System." Center for Advanced Computation Document no. 10. Urbana: University of Illinois, July 1971.

Door, Fred W. "The Direct Solution of the Discrete Poisson Equation on a Rectangle." SIAM Review, vol. 12, no.2 (1970): 248-263.

Ericksen, James H. "A Survey of Iterative Methods for Solving Poisson's Equation and Their Adaptibility to ILLIAC IV." Master's thesis. Urbana: University of Illinois, 1972.

Forsythe, G. E. and Wason, W. R. Finite-Difference Methods for Partial Differential Equations. New York: John Wiley and Sons, Inc., 1960.

Hockney, R. W. "The Potential Calculation and Some Applications." Methods in Computational Physics, 9 (1970): 136-211.

"ILLIAC IV Systems Characteristics and Programming Manual." Burroughs Corporation Document no. 66000B, Paoli, Pennsylvania, 1969.

Lawrie, D. H. "GLYPNIR Programming Manual." ILLIAC IV Document no. 232, Department of Computer Science. Urbana: University of Illinois, August 1970.

Ogura, M., M. S. Sher, and J. H. Ericksen. "A Study of the Efficiency of ILLIAC IV in Hydrodynamic Calculations." Center for Advanced Computation Document no. 59. Urbana: University of Illinois, December 1972.

Rudsinski, Lawrence E. "Tranquil Code for the MSOR Method for Solving LaPlace's Difference Equation." ILLIAC IV Document no. 208. Urbana: University of Illinois, December 1968.

Stevens, James E., Jr. "A Fast Fourier Transform Subroutine for ILLIAC IV." Center for Advanced Computation Document no. 17. Urbana: University of Illinois, July 1970.

Tod, J. Survey of Numerical Analysis. New York: 1962.

Wachspress, E. L. Iterative Solution of Elliptic Systems and Applications to the Neutron Diffusion Equations of Reactor Physics. Englewood Cliffs, New Jersey: 1966.

Widlund, Olof B. "On the Use of Fast Methods for Separable Fintie Difference Equations for the Solution of General Elliptic Problems." New York: Courant Institute of Mathematical Sciences.

Young, D. M. Iterative Solution of Large Linear Systems. New York: Academic Press, 1971.

APPENDIX A

TIMING METHODOLOGY

The procedure for estimating the amount of time required by an algorithm is explained below. Care has been taken to insure that the method of calculating the computer time does not give one algorithm an unwarranted advantage over another.

In determining the amount of time an algorithm takes on a computer we must consider the language in which the algorithm is programmed. In comparing GLYPNIR to ASK, one finds that the average GLYPNIR code takes about twice as long as the ASK code for the same problem. Two major reasons for this difference are memory fetching and indexing. The code generated by GLYPNIR for indexing can be improved considerably by rewriting it in ASK. GLYPNIR does fetching and storing of data between PEM and PE registers that can be eliminated by efficient use of the PE registers in ASK.

GLYPNIR code is much easier to write than ASK. One must choose between simplicity and speed. However, GLYPNIR permits the programmer to write sections of code in ASK. Thus the programmer can write code in GLYPNIR and then rewrite the statements which are executed most often in ASK. In estimating the time for the different methods, we have timed the statements which are executed in the inner loops of the iterative method. Since these statements take most of the computer time they should be written in ASK. We use the GLYPNIR statements as an outline of the method but the time estimates reflect efficient ASK code, not the code that GLYPNIR would generate. To show the reader how we arrived at our time estimates we have included a table with assignment statements, operations which are not overlapped and the clocks required for each method.

The assignment statements are written in a combination of GLYPNIR and ASK. The parts of the statements which are not used in GLYPNIR are defined in Table 14. A summary of the assumptions used in obtaining the time estimates is found in Table 15.

NOTATION	MEANING	NOTATION	MEANING
$A	PE Register A	$Ci	ACAR i for i=0,1,2,3
$B	PE Register B	U(i)	U indexed by ACAR i
$R	PE Register R	U[$X]	U indexed by register X
$S	PE Register S	U[$X](i)	both $X and $Ci indexing of U
$X	PE Register X	$Di	ADB storage location i for i=0,...,63

Table 14. Assignment Statement Notation

Every iterative algorithm we consider in this study will have two parts: Phase 1 will be used to improve 64 points simultaneously; Phase 2 will improve up to 63 points simultaneously. The machine efficiency of Phase 1 is greater than the machine efficiency of Phase 2. If one used only Phase 1 on certain meshes the extra time needed to bring the data to the PE's would exceed the time saved by improving machine efficiency. Thus the fastest code is obtained by a combination of Phase 1 and Phase 2.

1) CU instructions are overlapped completely by a combination of FINST/PE instructions and PEM fetches.

2) Memory fetching is overlapped as much as possible.

3) Seven PE clocks are used to load (store) a PE register from (to) PEM.

4) One PE clock issued to load a PE register from the CU.

5) Transfer between PE registers require one clock.

6) Each GRABONE will be considered to be an ASK LOAD instruction (10 clocks).

Table 15. Assumptions in Timing*

* The author feels these timing assumptions should give times within 20% of the actual times the algorithms would take on ILLIAC IV.

To aid in comparing times of different methods we will define an execution of Phase 1 to be the improvement of 64 points simultaneously by Phase 1. Similarly, an execution of Phase 2 is the improvement of k points where Phase 2 improves k points simultaneously.

APPENDIX B

MAJOR STEPS OF FACR AND MFACR FOR DIRICHLET'S BOUNDARY CONDITIONS

Given the meshes Ψ and U with mesh points at locations $\Psi_{i,j}$ and $U_{i,j}$ where $0 \le i \le M$ and $0 \le j \le N$ the five point difference scheme approximations of Poisson's equation becomes

$$\frac{U_{s-1,t} - 2U_{s,t} + U_{s+1,t}}{h_x^2} + \frac{U_{s,t-1} - 2U_{s,t} + U_{s,t+1}}{h_y^2} = \Psi_{s,t} \qquad (B.1)$$

To solve (B.1) FACR works as follows.

Odd/even reduction (see Section 3.1) calculates ψ^* for the even columns of Ψ. Equation (41) from Section 3.1 can be expanded into the following equation for individual mesh points.

$$\frac{1}{h_y^2}(U_{i,t-2} + U_{i,t+2}) - \frac{h_y^2}{h_x^4}(U_{i-2,t} + U_{i+2,t})$$

$$+ 4\left(\frac{h_y^2}{h_x^4} + \frac{1}{h_x^2}\right)(U_{i-1,t} + U_{i+1,t}) - \left(6\frac{h_y^2}{h_x^4} + \frac{8}{h_x^2} + \frac{2}{h_y^2}\right)U_{i,t} = \psi^*_{i,t} \qquad (B.2)$$

where

$$\psi^*_{i,t} = \Psi_{i,t-1} - \frac{h_y^2}{h_x^2}(\Psi_{i-1,t} + \Psi_{i+1,t}) + 2\left(\frac{h_y^2}{h_x^2} + 1\right)\Psi_{i,t} + \Psi_{i,t-1} \qquad (B.2.1)$$

Fourier analysis calculates the Fourier harmonics Ψ^s and Ψ^* defined by

$$\Psi^s_{i,t} = \frac{2}{M}\sum_{k=1}^{M-1} \Psi^*_{k,t} \,\mathrm{SIN}\, \frac{\pi k i}{M} . \qquad (B.3)$$

Combining (B.2) and (B.3) we obtain

$$\Psi^s_{i,t} = \frac{2}{M}\sum_{k=1}^{M-1} \left[\mathrm{SIN}\, \frac{\pi k i}{M} \left\{\frac{1}{h_y^2}(U_{k,t-2} + U_{k,t+2}) - \frac{h_y^2}{h_x^4}(U_{k-2,t} + U_{k+2,t})\right.\right.$$

$$\left.\left. + 4\left(\frac{h_y^2}{h_x^4} + \frac{1}{h_x^2}\right)(U_{k-1,t} + U_{k+1,t}) - \left(6\frac{h_y^2}{h_x^4} + \frac{8}{h_x^2} + \frac{2}{h_x^2}\right)U_{k,t}\right\}\right] \qquad (B.4)$$

The Fourier harmonics for U, U^s are defined as follows:

$$U^s_{i,t} = \frac{2}{M} \sum_{k=1}^{M-1} \text{SIN} \frac{\pi k i}{M} U_{k,t} \tag{B.5}$$

and thus

$$U_{k,t} = \sum_{\ell=1}^{M-1} \text{SIN} \frac{\pi \ell k}{M} U^s_{\ell,t} \ . \tag{B.6}$$

Combining (B.5), (B.6) and (B.4) we obtain

$$\Psi^s_{i,t} = \frac{1}{h_y^2} (U^s_{i,t+2} + U^s_{i,t-2}) - \left(6 \frac{h_y^2}{h_x^4} + \frac{8}{h_x^2} + \frac{2}{h_y^2} \right) U^s_{i,t}$$

$$+ \frac{2}{M} \sum_{k=1}^{M-1} \left[\text{SIN} \frac{\pi k i}{M} \left\{ - \frac{h_y^2}{h_x^4} \sum_{\ell=1}^{M-1} (\text{SIN} \frac{\pi(k-2)\ell}{M} U^s_{\ell,t} + \text{SIN} \frac{\pi(k+2)\ell}{M} U^s_{\ell,t}) \right.\right.$$

$$\left.\left. + 4 \left(\frac{h_y^2}{h_x^4} + \frac{1}{h_y^2} \right) \sum_{\ell=1}^{M} (\text{SIN} \frac{\pi(k-1)\ell}{M} U^s_{\ell,t} + \text{SIN} \frac{\pi(k+1)\ell}{M} U^s_{\ell,t}) \right\} \right] . \tag{B.7}$$

Using the facts that $\text{Sin}(a \pm b) = \text{Sin } a \text{ Cos } b \pm \text{Sin } b \text{ Cos } a$ and

$\sum_{k=1}^{M-1} \text{Sin} \frac{\pi \ell k}{M} \text{Sin} \frac{\pi i k}{M} = \frac{N}{2} \delta_{i,\ell}$ for $\ell, i = 1, 2, \ldots, M-1$ where

$\delta_{i,} = \begin{cases} 0 \text{ if } i \neq \ell \\ 1 \text{ if } i = \ell \end{cases}$ we can decouple (B.7) into a tridiagonal linear system

$$\Psi^s_{i,t} = \frac{1}{h_y^2} (U^s_{i,t+2} + U^s_{i,t-2}) - \left(2 \frac{h_y^2}{h_x^4} \text{COS} \frac{2\pi i}{M} - 8 \left(\frac{h_y^2}{h_x^4} + \frac{1}{h_x^2} \right) \text{COS} \frac{\pi i}{M} \right.$$

$$\left. + 6 \frac{h_y^2}{h_x^4} + \frac{8}{h_x^2} + \frac{2}{h_y^2} \right) U^s_{i,t} \tag{B.8}$$

CRED solves (B.8) for U^s.

Fourier synthesis calculates U from U^s using (B.6).

This gives us the values for U on the even columns.

Using the values of U on the even columns we can solve (B.1) for the values of U on the odd columns using a method which solves tridiagonal linear systems.

In summary FACR performs the following algorithm:

1. Given Ψ compute Ψ^* for the even columns using (B.2.1) and overwrite Ψ with Ψ^* on the even columns.
2. Using Ψ^* compute Ψ^s according to (B.3) and overwrite Ψ^* with Ψ^s.
3. Using Ψ^s solve (B.8) for U^s and overwrite Ψ^s with U^s.
4. Using U^s compute U according to (B.6) on the even columns and overwrite U^s with U on the even columns.
5. Using the values of U on the even columns solve (B.1) for the values of U on the odd columns and overwrite Ψ with U on the odd columns.

MFACR is similar to FACR but the odd/even reduction step is omitted. MFACR begins with Fourier analysis which calculates the Fourier harmonics Ψ^s for Ψ using

$$\Psi^s_{i,t} = \frac{2}{M} \sum_{k=1}^{M-1} \Psi_{k,t} \, \mathrm{SIN} \, \frac{\pi k i}{M} \tag{B.9}$$

From B.9 we obtain

$$\Psi^s_{i,t} = \frac{2}{M} \sum_{k=1}^{M-1} \mathrm{SIN} \, \frac{\pi k i}{M} \left(\frac{U_{s-1,t} - 2U_{s,t} + U_{s+1,t}}{h_x^2} + \frac{U_{s,t-1} - 2U_{s,t} + U_{s,t+1}}{h_y^2} \right) \tag{B.10}$$

Using the same methods employed to obtain (B.8) from (B.7)we obtain (B.11) from (B.10).

$$\psi^{s}_{i,t} = \frac{1}{h_y^2}\left(U^{s}_{i,t+1} + U^{s}_{i,t-1}\right) + \left(\frac{2}{h_y^2}\cos\frac{\pi i}{M} - \frac{2}{h_y^2} - \frac{2}{h_x^2}\right)U^{s}_{i,t} \qquad \text{(B.11)}$$

CRED solves (B.11) for $U^{s}_{i,t}$.

Finally (B.1) is used to calculate U for the entire mesh.

I L L I A C I V G L Y P N I R C O M P I L E R .

(B6500 VERSION 3.3.00)

```
 1  SUBROUTINE MSOR(PCPOINT OUT X,CREAL WB,CINT CN,CINT CM,CREAL BOUND,          00000100  00:00:000000
 2 PCPOINT ETA,CREAL DX, CREAL DZ, CINT OUT IT);                                00000200  01:00:000012
 3 %                                                                            00000300  01:00:000012
 4 %%%%%%%%%%%%%%%%%%% DEFINITION OF PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%  00000400  01:00:000012
 5 %                                                                            00000500  01:00:000012
 6 %   X IS THE MESH TO BE IMPROVED                                            00000600  01:00:000012
 7 %   WB IS THE OPTIMAL RELAXATION PARAMETER FOR SOR                           00000700  01:00:000012
 8 %   WB IS SUPPLIED BY THE MAIN PROGRAM                                       00000800  01:00:000012
 9 %   CN IS THE NUMBER OF ROWS IN THE MESH X                                   00000900  01:00:000012
10 %   CM IS THE NUMBER OF COLUMNS IN THE MESH X                                00001000  01:00:000012
11 %   BOUND IS THE MAX ERROR AT WHICH ANOTHER ITERATION WILL                   00001100  01:00:000012
12 %   NOT BE PERFORMED                                                         00001200  01:00:000012
13 %   ETA IS THE CONSTANT MESH IN THE EQUATION AX=ETA OBTAINED                 00001300  01:00:000012
14 %   FROM THE POISSON'S EQUATION                                              00001400  01:00:000012
15 %   DX IS THE DISTANCE BETWEEN NEIGHBORING COLUMNS                           00001500  01:00:000012
16 %   DZ IS THE DISTANCE BETWEEN NEIGHBORING ROWS                              00001600  01:00:000012
17 %   IT IS THE MAXIMUM NUMBER OF ITERATION WHICH WILL BE PERFORMED            00001700  01:00:000012
18 %   WHEN MSOR RETURNS CONTROL TO THE MAIN PROGRAM IT WILL                    00001800  01:00:000012
19 %   EQUAL THE NUMBER OF ITERATIONS PERFORMED                                 00001900  01:00:000012
20 %                                                                            00002000  01:00:000012
21 %%%%%%%%%  THE  USE  %%%%%% SOLVE POISSON EQUATION  %%%%%%%%%              00002100  01:00:000012
22 %                                                                            00002200  01:00:000012
23 %     CONSIDER A RECTANGULAR MESH,X[*],WHICH HAS N ROWS AND M COLUMNS.       00002300  01:00:000012
24 %     THE VALUES OF THE BOUNDARY ARE GIVEN.  THAT IS THE VALUES OF ROW       00002400  01:00:000012
25 %     1, ROW N, COLUMN 1 AND COLUMN M ARE KNOWN.                             00002500  01:00:000012
26 %     WE WANT TO OBTAIN INTERIOR VALUES SUCH THAT                            00002600  01:00:000012
27 %     X[I,J]=H*(X[I-1,J]+X[I+1,J])+V*(X[I,J+1]+X[I,J-1])-G*ETA[I,J]          00002700  01:00:000012
28 %     WHERE A=DY*DY,B=DX*DX,H=A/(2*(A+B)),V=B/(2*(A+B)),G=H*B                00002800  01:00:000012
29 %                                                                            00002900  01:00:000012
30 %%%%%%%%%  THE METHOD  %%%% MODIFIED SUCCESSIVE OVER-RELAXATION %%%%%      00003000  01:00:000012
31 %                                YOUNG'S METHOD [1]                         00003100  01:00:000012
32 %                                                                            00003200  01:00:000012
33 %     FIRST DIVIDE THE MESH POINTS INTO TWO SETS. THE FIRST SET CONTAINS     00003300  01:00:000012
34 %     ODD POINTS AND THE SECOND SET CONTAINS THE EVEN POINTS. X[I,J] IS      00003400  01:00:000012
35 %     ODD IF I+J IS ODD. IF I+J IS EVEN. THEN X[I,J] IS EVEN.                00003500  01:00:000012
36 %     EACH ITERATION IS DIVIDED INTO TWO PARTS. THE ODD PART AND THE         00003600  01:00:000012
37 %     EVEN PART.  THE ODD PART IS DONE FIRST.  THE ODD FORMULA IS            00003700  01:00:000012
38 %      (L+1)          (L)          (L)          (L)          (L)             00003800  01:00:000012
39 %    X[I,J]=W*(H*(X[I+1,J]+X[I-1,J])+V*(X[I,J+1]+X[I,J-1])-G*ETA[I,J])+     00003900  01:00:000012
40 %             (L)          (L)                                              00004000  01:00:000012
41 %     (W-1)*X[I,J] WHERE X[I,J] STANDS FOR X[I,J] AFTER L ITERATIONS.        00004100  01:00:000012
42 %     THE EVEN FORMULA IS                                                    00004200  01:00:000012
43 %     (L+1)          (L+1)        (L+1)          (L+1)        (L+1)          00004300  01:00:000012
44 %    X[I,J]=W*(H*(X[I+1,J]+X[I-1,J])+V*(X[I,J+1]+X[I,J-1])-G*ETA[I,J])+     00004400  01:00:000012
45 %             (L)                                                            00004500  01:00:000012
46 %     (W-1)*X[I,J]   IN BOTH FORMULAS W IS THE RELAXATION PARAMETER.         00004600  01:00:000012
47 %                                                                            00004700  01:00:000012
48 %%%%%%%%% THE ALGORITHM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00004800  01:00:000012
49 %                                                                            00004900  01:00:000012
50 %     THIS ALGORITHM HAS TWO PHASES  PHASE1 IMPROVES 64 POINTS AT A TIME     00005000  01:00:000012
```

```
51 %     UNTIL ALL THE POINTS IN CLASS ONE ARE IMPROVED.  PHASE2 IMPROVES
52 %     ALL THE POINTS IN CLASS TWO, TWO ROWS AT A TIME.
53 %     CONSIDER THE MESH X[ * ] WHERE FOR 1<I<N AND 1<J<M X[I,J] IS TO BE
54 %     IMPROVED.  DIVIDE THESE POINTS INTO TWO CLASSES.
55 %     CLASS ONE CONSISTS OF X[I,J] SUCH THAT 1<I<N AND I<J<M-(M-2)MOD 64
56 %     CLASS TWO CONSISTS OF X[I,J] SUCH THAT 1<I<N AND(M-1-(M-2)MOD 64)<
57 %     J<M.  TO IMPLIMENT THIS I USE STRAIGHT ROW WRAPAROUND STORAGE.
58 %
59 %%%%%%%%%%%%%%%  CODE NAME:  MSOR   %%%%%%%%%%%%%%%%%%%%
60 %
61 %%%%%%%%%%%%%%%    FUNCTION:   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
62 %
63 %     SOLVES POISSONS EQUATION
64 %
65 %%%%%%%%%%%%%   LANGUAGE: GLYPNIR  %%%%%%%%%%%%%%%%%%
66 %
67 %%%%%%%%%%%%%%%%%%%%  REFERENCES  %%%%%%%%%%%%%%%%%%%%%%%%%%%%
68 %
69 %       [1] SECTION 2.3 OF THIS PAPER
70 %       [2] VARGA, R. S., "MATRIX ITERATIVE ANALYSIS",
71 %           PRENTICE-HALL,ENGLEWOOD CLIFFS, NEW JERSEY(1962).
72 %       [3] LAWRIE, D. H., "GLYPNIR PROGRAMMING MANUAL",
73 %           ILLIAC IV DOCUMENT NO. 232, DEPARTMENT OF COMPUTER
74 %           SCIENCE, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN,
75 %           URBANA, ILLINOIS(AUGUST, 1970).
76 %
77 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78 %
79       BEGIN
80        BOOLEAN DMODE,OLDMODE;
81        CINT G,CPI;
82        CINT    II,CNM2,CNM3,CM1,CMIK,CI,CCN,CODD,CJ,CK;
83        CINT C;
84        CREAL ERAS,OMEGA1,ALPH ,BETA,GI;
85        CREAL W,WI,AL,BE;
86        CINT Y;
87        PINT NP;
88        PINT JNI;
89        PINT PJ,JI,JIN,PN;
90        PINT PN2,JP;
91        LABEL DONE;
92        W:=WB;
93        WI:=1-WB;
94        CM:=CM-I;
95        CMI:=CM.[16:42];
96        BE:=DX*DX;
97        AL:=DZ*DZ;
98        ERAS:=0.5/(BE+AL);
99        GI:=W*AL*BE/(2.0*(AL+BE));
100       ALPH:=1.0/BE;
101       BETA:=1.0/AL;
102       CPI:=CMI+1;
103 %     Y IS THE NUMBER OF ROWS OF PE MEMORY IT TAKES TO STORE ONE ROW OF
104 %     A CN BY CM MESH.
105       Y:=CPI;
106       G:=2*CPI;
107 %     IF CODD EQUALS ONE WE HAVE AN ODD NUMBER OF ROWS TO IMPROVE.
108       CODD:=CN.[63:1];
109 %     CMIK IS USED TO HANDLE CASE WHERE PHASE 2 MUST IMPROVE
```

```
00005100  01:00:000012
00005200  01:00:000012
00005300  01:00:000012
00005400  01:00:000012
00005500  01:00:000012
00005600  01:00:000012
00005700  01:00:000012
00005800  01:00:000012
00005900  01:00:000012
00006000  01:00:000012
00006100  01:00:000012
00006200  01:00:000012
00006300  01:00:000012
00006400  01:00:000012
00006500  01:00:000012
00006600  01:00:000012
00006700  01:00:000012
00006800  01:00:000012
00006900  01:00:000012
00007000  01:00:000012
00007100  01:00:000012
00007200  01:00:000012
00007300  01:00:000012
00007400  01:00:000012
00007500  01:00:000012
00007600  01:00:000012
00007700  01:00:000012
00007800  01:00:000012
00007850  01:00:000012
00007900  02:00:000012
00008000  02:00:000012
00008100  02:00:000012
00008200  02:00:000012
00008300  02:00:000012
00008400  02:00:000012
00008500  02:00:000012
00008600  02:00:000012
00008700  02:00:000012
00008800  02:00:000012
00008900  02:00:000012
00009000  02:00:000012
00009100  02:00:000012
00009200  02:00:000016
00009300  02:00:000028
00009400  02:00:000035
00009500  02:00:000041
00009600  02:00:000050
00009700  02:00:000059
00009800  02:00:000077
00009900  02:00:000106
00010000  02:00:000117
00010100  02:00:000128
00010200  02:00:000135
00010300  02:00:000135
00010400  02:00:000135
00010500  02:00:000139
00010600  02:00:000149
00010700  02:00:000149
00010800  02:00:000155
00010900  02:00:000155
```

```
111       CMIK:=CM.[58:6];                                                    00011000  02:00:000155
112       IF CMIK=0 THEN BEGIN                                                00011100  02:00:000161
113       CMI:=CMI-1;CMIK:=64;                                                00011200  03:01:000178
114       END ELSE IF CMIK=1 THEN CMIK=0;                                     00011300  03:01:000190
                                                                   00011100
115       CNM3:=CN-3;                                                         00011400  02:00:000225
116       CNM3:=CNM3*CPI;                                                     00011500  02:00:000232
117 %     PJ,NP,PN,CCN AND CNM2 ARE USED TO HELP EACH PE ACCESS THE RIGHT     00011600  02:00:000244
118 %     MEMORY LOCATION, THAT IS OBTAIN THE FOUR NEIGHBORS IN THE FIVE      00011700  02:00:000244
119 %     POINT DIFFERENCE EQUATION.                                          00011800  02:00:000244
120       PJ:=0;PN:=0;                                                        00011900  02:00:000244
121       MODE:=BOOLEAN(8000000000000000(16));                                00012000  02:00:000248
122       PN:=1;                                                              00012100  02:00:000253
123       MODE:=BOOLEAN(4000000000000000(16));                                00012200  02:00:000255
124       NP:=1;                                                              00012300  02:00:000260
125       CNM2:=CN-2;                                                         00012400  02:00:000262
126       CNM2:=CNM2*CPI;                                                     00012500  02:00:000269
127       MODE:=BOOLEAN(5555555555555555(16));                                00012600  02:00:000281
128       PJ:=CPI;                                                            00012700  02:00:000285
129       OLDMODE:= BOOLEAN(7FFFFFFFFFFFFFFF(16));                            00012800  02:00:000290
130       MODE:=TRUE;                                                         00012900  02:00:000295
131       CCN:=CM1-1;                                                         00013000  02:00:000299
132 %     THIS LOOP IMPROVES ALL MESH POINTS. THUS EACH TIME IT IS EXECUTED   00013100  02:00:000306
133 %     WE INCREASE THE ITERATION COUNTER BY ONE.                           00013200  02:00:000306
134       LOOP II:=2,1,IT DO BEGIN                                            00013300  02:00:000306
135 %     THE ITERATION IS DIVIDED INTO TWO PARTS                             00013400  03:00:000330
136 %     IF THERE ARE M POINTS TO BE IMPROVED IN EACH ROW WHERE M=64N+I,     00013500  03:00:000330
137 %     N AND I ARE INTEGERS AND I<64 THEN PART ONE IMPROVES THE FIRST 64N  00013600  03:00:000330
138 %     POINTS AND PART TWO IMPROVES THE REMAINING I POINTS ON EACH ROW.    00013700  03:00:000330
139       C:=0;                                                               00013800  03:00:000330
140       LOOP CK:=1,1,2 DO BEGIN                                             00013900  03:00:000335
141 %     PART ONE IMPROVES 64 POINTS AT A TIME IN TWO ADJACENT ROWS.         00014000  04:00:000346
142       LOOP CJ:=0,1,CCN DO BEGIN                                           00014100  04:00:000346
143       JI:=PJ+CPI;                                                         00014200  05:00:000370
144       JN1:=[1]JI;                                                         00014300  05:00:000375
145       JIN:=JN1+NP+PN;                                                     00014400  05:00:000405
146       J1:=J1+PN;                                                          00014500  05:00:000409
147 %     THIS LOOP IMPROVES LINES TWO AT A TIME THUS IF THERE IS AN ODD      00014600  05:00:000412
148 %     NUMBER OF LINES WE MUST DEAL WITH THE LAST LINE IN A SPECIAL WAY.   00014700  05:00:000412
149       LOOP CI:=CPI,G,CNM3 DO BEGIN                                        00014800  05:00:000412
150       PN2:=X[JI];                                                         00014900  06:00:000453
151       X[JI]:=GI*((X[JI+Y]+X[JI-Y])*ALPH+BETA*(RTL(1,,X[JIN])              00015000  06:00:000460
152       +RTR(1,,X[JN1]))- ETA[JI])+W1*X[JI];                                00015100  06:00:000491
153       JI:=JI+G;                                                           00015200  06:00:000525
154       JN1:=JN1+G;                                                         00015300  06:00:000530
155       JIN:=JIN+G;                                                         00015400  06:00:000535
156       IF ABS(PN2-X[JI]) GEQ BOUND THEN C:=1;                              00015500  06:00:000540
157       END;                                                                00015600  06:00:000575
                                                                   00014800
158 %     IF WE HAVE AN ODD NUMBER OF LINES THE LAST LINE IS IMPROVED HERE.   00015700  05:00:000583
159       IF CODD=1 THEN BEGIN                                                00015800  05:00:000583
160       JN1:=CNM2+CJ;                                                       00015900  06:01:000600
161       JI:=JN1+PN;                                                         00016000  06:01:000606
162       JIN:=JI+NP;                                                         00016100  06:01:000609
163       PN2:=X[JI];                                                         00016200  06:01:000612
164       X[JI]:=GI*((X[JI+Y]+X[JI-Y])*ALPH+BETA*(RTL(1,,X[JIN])              00016300  06:01:000619
165       +RTR(1,,X[JN1]))- ETA[JI])+W1*X[JI];                                00016400  06:01:000650
166       IF ABS(PN2-X[JI]) GEQ BOUND THEN C:=1;                              00016500  06:01:000684
167       END;                                                                00016600  06:01:000719
                                                                   00015800
```

```
168         PJ:=PJ+1;                                                                    00016700  05:00:000722
169         END;                                                                         00016800  05:00:000725
                                                                                 00014100
170         JI:=PJ;                                                                      00016900  04:00:000733
171         JNI:=[!]JI;                                                                  00017000  04:00:000735
172 %       PART TWO IMPROVES THE REMAINING POINTS.                                      00017100  04:00:000765
173         IF PEN<CMIN AND OLDMODE THEN BEGIN                                           00017200  04:00:000765
174         DMODE:=MODE;                                                                 00017300  05:01:000786
175 %       THIS LOOP IMPROVES TWO LINES AT A TIME.                                      00017400  05:01:000788
176         LOOP CI:=CPI+G+CNM3 DO BEGIN                                                 00017500  05:01:000788
177         MODE:=TRUE;                                                                  00017600  06:01:000829
178         JI:=JI+CI;                                                                   00017700  06:01:000833
179         JNI:=JNI+CI;                                                                 00017800  06:01:000837
180         JIN:=JNI+PN;                                                                 00017900  06:01:000841
181         MODE:=DMODE;                                                                 00018000  06:01:000844
182         PN2:=X[JI];                                                                  00018100  06:01:000848
183         X[JI]:=GI*((X[JI+Y]+X[JI-Y])*ALPH+BETA*(RTL(1,,X[JIN])                       00018200  06:01:000855
184         +RTR(1,,X[JNI]))- ETA[JI])+W1*X[JI];                                         00018300  06:01:000886
185         IF ABS(PN2-X[JI]) GEQ BOUND THEN C:=1;                                       00018400  06:01:000920
186         END;                                                                         00018500  06:01:000955
                                                                                 00017500
187         MODE:=TRUE;                                                                  00018600  05:01:000963
188         JNI:=CNM2+CM1;                                                               00018700  05:01:000967
189         JI:=JNI+PN;                                                                  00018800  05:01:000974
190         MODE:=DMODE;                                                                 00018900  05:01:000977
191 %       THIS LOOP IMPROVES THE LAST LINE IF THERE IS AN ODD NO. OF LINES.             00019000  05:01:000981
192         IF ODD=1                THEN BEGIN                                           00019100  05:01:000981
193         PN2:=X[JI];                                                                  00019200  06:02:000998
194         X[JI]:=GI*((X[JI+Y]+X[JI-Y])*ALPH+BETA*(RTL(1,,X[JI])                        00019300  06:02:001005
195         +RTR(1,,X[JNI]))- ETA[JI])+W1*X[JI];                                         00019400  06:02:001036
196         IF ABS(PN2-X[JI]) GEQ BOUND THEN C:=1;                                       00019500  06:02:001070
197         END;                                                                         00019600  06:02:001105
                                                                                 00019100
198         END;                                                                         00019700  05:01:001108
                                                                                 00017200
199         PJ:=[!]PJ-CM1;                                                               00019800  04:00:001111
200         END;                                                                         00019900  04:00:001144
                                                                                 00013900
201         IF C=0 THEN GO TO DONE;                                                      00020000  03:00:001152
202         END;                                                                         00020100  03:00:001176
                                                                                 00013300
203 DONE:   IT:=I1;                                                                      00020200  02:00:001184
204         END.                                                                         00020300  02:00:001189
                                                                                 00007850
```

NUMBER OF ERRORS DETECTED=0000. NUMBER OF WARNINGS NOTED=0000.
SOURCE PROGRAM SIZE= 00204 CARD IMAGES, 001637 SYNTACTIC ITEMS.
SOURCE FILE: CARD=ERICKSEN/PRINT/MSOR.
MACRO LIBRARY: ILLIAC/GLYPNIR3/MACROS.
ESTIMATED CORE MEMORY REQUIREMENTS= 000717 WORDS CODE, 000868 WORDS DATA.
COMPILATION TIME= 0018 SECONDS ELAPSED, 0009 SECONDS PROCESSING.

ILLIAC IV GLYPNIR COMPILER

(B6500 VERSION 3.3.00)

```
 1       SUBROUTINE MSLOR(PCPOINT OUT U,CREAL W,CINT CN,CINT CM,CREAL          00000100  00:00:000000
 2   BOUND,PCPOINT T,CREAL DX,CREAL DZ,CINT OUT IT);                          00000200  01:00:000012
 3   %                                                                        00000300  01:00:000012
 4   %%%%%%%%%%%%% DEFINITION OF PARAMETERS %%%%%%%%%%%%%%%%%               00000400  01:00:000012
 5   %                                                                        00000500  01:00:000012
 6   %    U IS THE MESH TO BE EVALUATED IN POISSON'S EQUATION                 00000600  01:00:000012
 7   %    T IS THE CONSTANT MESH                                              00000700  01:00:000012
 8   %    W IS THE RELAXATION PARAMETER WHICH IS SUPLIED BY THE               00000800  01:00:000012
 9   %    MAIN PROGRAM                                                        00000900  01:00:000012
10   %    CM IS THE NUMBER OF COLUMNS IN U                                    00001000  01:00:000012
11   %    CN IS THE NUMBER OF ROWS IN U                                       00001100  01:00:000012
12   %    DX IS THE DISTANCE BETWEEN ROWS IN U                                00001200  01:00:000012
13   %    DY IS THE DISTANCE BETWEEN COLUMNS IN U                             00001300  01:00:000012
14   %    IT ENTERS AS THE MAXIMUM NUMBER OF ITERATIONS TO BE                 00001400  01:00:000012
15   %    PERFORMED AND IS RETURNED TO THE MAIN PROGRAM AS THE                00001500  01:00:000012
16   %    NUMBER OF ITERATIONS PERFORMED                                      00001600  01:00:000012
17   %                                                                        00001700  01:00:000012
18   %%%%%%%%%  THE USE   %%%%%%%%   SOLVES POISSON'S EQUATIONS              00001800  01:00:000012
19   %                                                                        00001900  01:00:000012
20   %    CONSIDER A RECTANGULAR MESH, U[*], WHICH HAS CN ROWS AND            00002000  01:00:000012
21   %    CM COLUMNS.  THE VALUES OF THE BOUNDARY ARE GIVEN.  THAT IS         00002100  01:00:000012
22   %    THE VALUES OF ROW 1, ROW CN, COLUMN 1, AND COLUMN CM ARE            00002200  01:00:000012
23   %    KNOWN.  THE CONSTANT MESH T IS ALSO KNOWN. WE WANT TO OBTAIN        00002300  01:00:000012
24   %    THE INTERIOR POINTS OF U SUCH THAT:                                 00002400  01:00:000012
25   %    U[I,J]=H*(U[I-1,J]+U[I+1,J])+V*(U[I,J-1]+U[I,J+1])-G*T[I,J]         00002500  01:00:000012
26   %    WHERE A=DY*DY,B=DX*DX,H=A/(2*(A+B)),V=B/(2*(A+B)), AND G=H*B         00002600  01:00:000012
27   %    THIS PROGRAM IS WRITTEN TO WORK ON ANY RECTANGULAR MESH             00002700  01:00:000012
28   %    THAT FITS ON CORE.                                                  00002800  01:00:000012
29   %                                                                        00002900  01:00:000012
30   %%%%%   METHOD   %%%%%%   SUCCESSIVE LINE OVERRELAXATION   %%%%%        00003000  01:00:000012
31   %                                                                        00003100  01:00:000012
32   %    THIS PROGRAM USES THE CUTHILL-VARGA METHOD [3] PAGE 238 TO          00003200  01:00:000012
33   %    PERFORM COLUMN RELAXATION ON THE MESH U WHICH IS STORED IN          00003300  01:00:000012
34   %    ODD-EVEN STORAGE AS FOUND IN [1] PAGE 22.  T IS ALSO IN             00003400  01:00:000012
35   %    ODD-EVEN STORAGE.                                                   00003500  01:00:000012
36   %    THE COEFFICIENTS CALCULATED IN PRECONDIONING, [2] PAGE 32, ARE      00003600  01:00:000012
37   %    DISB,DI,D, AND E.                                                   00003700  01:00:000012
38   %                                                                        00003800  01:00:000012
39   %%%%%%%%%%%   CODE NAME: MSLOR   %%%%%%%%%%%%%%%%%%%%%%%%%%          00003900  01:00:000012
40   %                                                                        00004000  01:00:000012
41   %%%%%%%%%%%%%%%%   FUNCTION:   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%     00004100  01:00:000012
42   %                                                                        00004200  01:00:000012
43   %    SOLVES POISSON'S EQUATION                                           00004300  01:00:000012
44   %                                                                        00004400  01:00:000012
45   %%%%%    LANGUAGE: GLYPNIR   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   00004500  01:00:000012
46   %                                                                        00004600  01:00:000012
47   %%%%%%%%%%%%%%%%%   REFERENCES   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   00004700  01:00:000012
48   %                                                                        00004800  01:00:000012
49   %    [1] SECTION 2.3.4 OF THIS PAPER                                     00004900  01:00:000012
50   %    [2] SECTION 2.4.4 OF THIS PAPER                                     00005000  01:00:000012
```

```
%   [3] CUTHILL AND VARGA, "A METHOD OF NORMALIZING BLOCK            00005100  01:00:000012
%       ITERATION", J. ASSOC. COMPUT. MACH., VOL. 6, PP. 236-         00005200  01:00:000012
%       244, 1959.                                                    00005300  01:00:000012
%   [4] LAWRIE, D. H., "GLYPNIR PROGRAMMING MANUAL",                  00005400  01:00:000012
%       ILLIAC IV DOCUMENT NO. 232, DEPARTMENT OF COMPUTER            00005500  01:00:000012
%       SCIENCE, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN,          00005600  01:00:000012
%       URBANA, ILLINOIS, 1970.                                       00005700  01:00:000012
%                                                                     00005800  01:00:000012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   00005900  01:00:000012
%                                                                     00006000  01:00:000012
  BEGIN                                                               00006100  01:00:000012
    PCPOINT DISB;                                                     00006200  02:00:000012
    CINT MC;                                                          00006300  02:00:000012
    PCPOINT STORE,E,D,DI;                                             00006400  02:00:000012
    CINT CM2,CM3,CN2K,OD,C,B,EV,CM4,LPR,CI,CJ,II,CP,LP,ODD,CN2;       00006500  02:00:000012
    PINT PN,LPN;                                                      00006600  02:00:000012
    PREAL ST;                                                         00006700  02:00:000012
    CINT I,K,KPKM1,JC;                                                00006800  02:00:000012
    CREAL ALPH,RABONE,BETA,GAMMA,ALPHS;                               00006900  02:00:000012
    BOOLEAN OLDMODE,DMODE;                                            00007000  02:00:000012
    LABEL DONE;                                                       00007100  02:00:000012
    CM2:=CM-2;%THE NUMBER OF COLUMNS TO BE IMPROVED                   00007200  02:00:000012
    STORE:=GETPEB(CM2); % TEMPORARY STORAGE NEEDED FOR OVERRELAXATION 00007300  02:00:000019
    LPR:=CM.[16:42]  ;% LINES OF PEM FOR STORING D,DI OR E MINUS ONE. 00007400  02:00:000027
    IF CM.[58:6] GEQ 3 THEN LPR:=LPR+1;                               00007500  02:00:000033
    D:=GETPEB(LPR);%DIAGONAL CONSTANTS USED BY CUTHILL + VARGA.       00007600  02:00:000063
    DISB:=GETPEB(LPR);                                                00007700  02:00:000071
    DI:=GETPEB(LPR);% INVERSE OF DI                                   00007800  02:00:000079
    E:=GETPEB(LPR);% OTHER CONSTANTS IN CUTHILL-VARGA METHOD.         00007900  02:00:000087
    LPR:=LPR-1;                                                       00008000  02:00:000095
    ALPH:=-1.0/(DZ*DZ); % ROW ELEMENTS OFF THE DIAGONAL               00008100  02:00:000102
    BETA:=-1.0/(DX*DX); % COLUMN ELEMENTS OFF THE DIAGONAL            00008200  02:00:000121
    GAMMA:=-2.0*(ALPH+BETA);%THE DIAGONAL ELEMENTS                    00008300  02:00:000140
    ALPHS:=ALPH*ALPH;                                                 00008400  02:00:000156
    DMODE:=BOOLEAN(8000000000000000(16));                             00008500  02:00:000165
    OLDMODE:=BOOLEAN(7FFFFFFFFFFFFFFF(16));                           00008600  02:00:000170
    CN2:=CN-2; % NUMBER OF ROWS TO BE IMPROVED                        00008700  02:00:000175
    CN2K:=CN2.[57:6];% NUMBER OF PES USED IN PHASE TWO                00008800  02:00:000182
    OD:=CN2.[16:41]; %(# OF SETS OF 64 ODD ROWS IMPROVED BY PHASE 1   00008900  02:00:000188
    IF CN.[57:7]=1 THEN K:=3 ELSE K:=2;                               00009000  02:00:000194
    K:=K+OD+OD;                                                       00009100  02:00:000233
    EV:=(CN-1).[16:41];% # OF SETS OF 64 EVEN ROWS PHASE 1 IMPROVED   00009200  02:00:000245
    KPKM1:=K+K-1;                                                     00009300  02:00:000254
    CM3:=CM-3;                                                        00009400  02:00:000264
    CM4:=CM-4;                                                        00009500  02:00:000271
    PN:=0;  % VARIABLE USED AS AN INDEX                               00009600  02:00:000278
    MODE:=BOOLEAN(1);                                                 00009700  02:00:000280
    PN:=1;                                                            00009800  02:00:000285
    MODE:=TRUE;                                                       00009900  02:00:000287
    IF PEN=0 THEN D[0]:=GAMMA;                                        00010000  02:00:000291
%   THE FOLLOWING SECTION CALCULATES D,DI AND E                       00010100  02:00:000314
    LOOP CI:=1,1,CM3 DO IF CI.[58:6]=PEN THEN BEGIN                   00010200  02:00:000314
    C:=CI.[16:42];D[C]:=GAMMA-ALPHS/ RTR(1,,D[C-PN]);END;             00010300  03:01:000356
                                                              00010200
    B:=CM2.[58:6];                                                    00010400  02:00:000401
    IF B NEQ 0 AND PEN GEQ B THEN D[LPR]:=1.0;                        00010500  02:00:000407
    LOOP CI:=0,1,LPR   DO BEGIN                                       00010600  02:00:000453
    D[CI]:=SQRT(D[CI]);                                               00010700  03:00:000477
    DISB[CI]:=D[CI]*DI[CI]*BETA;                                      00010800  03:00:000488
```

```
110       E[CI]:=ALPH*DI[CI];END;                                                    00011000  03:00:000517
                                                                                  00010600
111       MODE:=OLDMODE;                                                             00011100  02:00:000536
112       E[0]:=E[0]*RTR(1,,DI[0]);                                                  00011200  02:00:000540
113       MODE:=TRUE;                                                                00011300  02:00:000554
114       LOOP CI:=1,1,LPR   DO E[CI]:=E[CI]*RTR(1,,DI[CI-PN]);                      00011400  02:00:000558
115       ODD:=CN.[63:1];% IF ODD=1 THEN ROWS TO BE IMPROVED IS ODD.                 00011500  02:00:000613
116 %     PREPARE FIRST AND LAST COLUMN OF MESH FOR ITERATION                        00011600  02:00:000619
117       LPN:=K-1;                                                                  00011700  02:00:000619
118       IF PEN=0 THEN LPN:=0;                                                      00011800  02:00:000624
119       LOOP CI:=0,1,CM3 DO                                                        00011900  02:00:000644
120       BEGIN LPN:=LPN+K;                                                          00012000  02:00:000668
121       MC:=CI.[16:42];                                                            00012100  03:00:000673
122       RABONE:=GRABONE(D[MC],CI);                                                 00012200  03:00:000678
123       IF CN2K=0 THEN U[LPN+K-1]:=RABONE*U[LPN+K-1];                              00012300  03:00:000688
124       IF PEN=0 OR PEN=CN2K THEN                                                  00012400  03:00:000729
125       U[LPN]:=RABONE*U[LPN];END;                                                 00012500  03:00:000749
                                                                                  00012000
126       LP:=0;CJ:=CM-1;                                                            00012600  02:00:000776
127 %     ADD THE FIRST AND THE LAST ROW OF THE MESH TO T                            00012700  02:00:000788
128       C:= CM3*K;                                                                 00012800  02:00:000788
129       LOOP CI:=K,1,KPKM) DO BEGIN                                                00012900  02:00:000800
130       T[CI]:=T[CI]+ALPH*U[CI-K];                                                 00013000  03:00:000832
131       CP:=CI+C;                                                                  00013100  03:00:000854
132       T[CP]:=T[CP]+ALPH*U[CP+K];                                                 00013200  03:00:000862
133       END;                                                                       00013300  03:00:000887
                                                                                  00012900
134 %     MULTIPLY T BY D INVERSE, DI                                                00013400  02:00:000895
135       CP:=K;                                                                     00013500  02:00:000895
136       LOOP CI:=0,1,CM3 DO LOOP CJ:=0,1,K-1 DO BEGIN                              00013600  02:00:000899
137       II:=CI.[16:42];                                                            00013700  03:00:000950
138       T[CP]:=-T[CP]*GRABONE(DI[II],CI);                                          00013800  03:00:000955
139       CP:=CP+1;END;                                                              00013900  03:00:000976
                                                                                  00013600
140       LOOP II:=1,1,IT DO BEGIN % EACH TIME THIS LOOP CYCLES IS ONE ITER.         00014000  02:00:000999
141       B:=0;                                                                      00014100  03:00:001023
142 %     PHASE ONE STARTS HERE                                                      00014200  03:00:001028
143       LOOP CI:=1,1,EV  DO BEGIN                                                  00014300  03:00:001028
144 %     LPN ACCESSES THE ODD COLUMNS, LP ACCESSES THE EVEN COLUMNS                 00014400  04:00:001052
145 %     THIS PART OF PHASE 1 IMPROVES THE EVEN COLUMNS                             00014500  04:00:001052
146       LP:=CI+K;                                                                  00014600  04:00:001052
147       LPN:=LP-);                                                                 00014700  04:00:001060
148       MODE:=OMODE;LPN:=LPN+2;   MODE:=TRUE;                                      00014800  04:00:001065
149       STORE[0 ]:=T[LP]-GRABONE(DISB[0],0)*(RTL(1,,U[LPN])+U[LP-1]);              00014900  04:00:001076
150       LOOP CJ:=1,1,CM3 DO BEGIN C:=CJ.[16:42];                                   00015000  04:00:001115
151       LP:=LP+K;LPN:=LPN+K;                                                       00015100  05:00:001144
152       STORE[CJ]:=T[LP]-GRABONE(DISB[C],CJ)*(RTL(1,,U[LPN])+U[LP-1])-             00015200  05:00:001158
153         GRABONE(E[C],CJ)*STORE[CJ-1];                                            00015300  05:00:001196
154       END;                                                                       00015400  05:00:001221
                                                                                  00015000
155       LOOP MC:=1,1,CM3 DO BEGIN                                                  00015500  04:00:001229
156       CJ:=CM3-MC;                                                                00015600  05:00:001253
157       JC:=CJ+1;                                                                  00015700  05:00:001261
158       C:=JC.[16:42];                                                             00015800  05:00:001268
159       STORE[CJ]:=STORE[CJ]-GRABONE(E[C],JC)*STORE[JC];                           00015900  05:00:001274
160       END;                                                                       00016000  05:00:001301
                                                                                  00015500
161       LP:=CI+K;                                                                  00016100  04:00:001309
162       LOOP CJ:=0,1,CM3 DO BEGIN % PERFORM OVERRELAXATION                         00016200  04:00:001317
163       ST:=U[LP];                                                                 00016300  05:00:001341
```

```
164       U[LP]:=W*(STORE[CJ]-U[LP))+U[LP];                                    00016400  05:00:001347
165       IF ABS(U[LP]-ST)>BOUND THEN B:=1; % CHECK ERROR                      00016500  05:00:001372
166       LP:=LP+K;END;                                                        00016600  05:00:001404
                                                                            00016200
167       END;                                                                 00016700  04:00:001421
                                                                            00014300
168 %     PHASE TWO STARTS HERE                                                00016800  03:00:001429
169       IF CN2K+ODD>0 AND CN2K+ODD<64 THEN BEGIN                             00016900  03:00:001429
170       LP:=KPKM1;                                                           00017000  04:01:001466
171       LPN:=LP-1;                                                           00017100  04:01:001470
172       STORE[0 ]:=T[LP]-GRABONE(DISB[0],0)*(RTL(1,,U[LPN])+U[LP-1]);        00017200  04:01:001475
173       LOOP CJ:=1,1,CM3 DO BEGIN C:=CJ.[16:42];                             00017300  04:01:001514
174       LP:=LP+K;LPN:=LPN+K;                                                 00017400  05:01:001543
175       STORE[CJ]:=T[LP]-GRABONE(DISB[C],CJ)*(RTL(1,,U[LPN])+U[LP-1])-       00017500  05:01:001557
176         GRABONE(E[C],CJ)*STORE[CJ-1];                                      00017600  05:01:001595
177       END;                                                                 00017700  05:01:001620
                                                                            00017300
178       LOOP MC:=1,1,CM3 DO BEGIN                                            00017800  04:01:001628
179       CJ:=CM3-MC;                                                          00017900  05:01:001652
180       JC:=CJ+1;                                                            00018000  05:01:001660
181       C:=JC.[16:42];                                                       00018100  05:01:001667
182       STORE[CJ]:=STORE[CJ]-GRABONE(E[C],JC)*STORE[JC];                     00018200  05:01:001673
183       END;                                                                 00018300  05:01:001700
                                                                            00017800
184       LP:=KPKM1;                                                           00018400  04:01:001708
185       LOOP CJ:=0,1,CM3 DO BEGIN % PERFORM OVERRELAXATION                   00018500  04:01:001712
186       STORE[CJ]:=W*(STORE[CJ]-U[LP])+U[LP];                                00018600  05:01:001736
187       IF (ODD+CN2K) >   PEN THEN BEGIN                                     00018700  05:01:001760
188       IF ABS(U[LP]- STORE[CJ])>BOUND THEN B:=1; % CHECK ERROR              00018800  06:02:001781
189       U[LP]:=STORE[CJ];END;                                                00018900  06:02:001817
                                                                            00018700
190       JC:=CJ+1;                                                            00019000  05:01:001829
191       LP:=LP+K;END;                                                        00019100  05:01:001835
                                                                            00018500
192       END;                                                                 00019200  04:01:001852
                                                                            00016900
193 %     PHASE ONE STARTS HERE                                                00019300  03:00:001855
194 %     LNP WILL BE USED TO ACCESS THE ODD COLUMNS, LP THE EVEN               00019400  03:00:001855
195       LOOP CI:=1,1,OD  DO BEGIN                                            00019500  03:00:001855
196       LP:=C1+K;                                                            00019600  04:00:001879
197       LPN:=LP-1;                                                           00019700  04:00:001887
198       MODE:=DMODE;LPN:=LPN+2;   MODE:=TRUE;                                00019800  04:00:001892
199       STORE[0]:=T[LPN]-GRABONE(DISB[0],0)*(RTR(1,,U[LP])+U[LPN+1]);        00019900  04:00:001903
200       LOOP CJ:=1,1,CM3 DO BEGIN C:=CJ.[16:42];                             00020000  04:00:001938
201       LP:=LP+K;LPN:=LPN+K;                                                 00020100  05:00:001967
202       STORE[CJ]:=T[LPN]-GRABONE(DISB[C],CJ)*(RTR(1,,U[LP])+U[LPN+1])-      00020200  05:00:001981
203         GRABONE(E[C],CJ)*STORE[CJ-1];                                      00020300  05:00:002015
204       END;                                                                 00020400  05:00:002040
                                                                            00020000
205       LOOP MC:=1,1,CM3 DO BEGIN                                            00020500  04:00:002048
206       CJ:=CM3-MC;                                                          00020600  05:00:002072
207       JC:=CJ+1;                                                            00020700  05:00:002080
208       C:=JC.[16:42];                                                       00020800  05:00:002087
209       STORE[CJ]:=STORE[CJ]-GRABONE(E[C],JC)*STORE[JC];                     00020900  05:00:002093
210       END;                                                                 00021000  05:00:002120
                                                                            00020500
211       LPN:=K+CI-1;                                                         00021100  04:00:002128
212       MODE:=DMODE;LPN:=LPN+2;   MODE:=TRUE;                                00021200  04:00:002135
213       LOOP CJ:=0,1,CM3 DO BEGIN % PERFORM OVERRELAXATION                   00021300  04:00:002146
```

```
215       U[LPN]:=W*(STORE[CJ]-U[LPN])+U[LPN];                                                  00021500  05:00:002174
216       IF ABS(U[LPN]-ST)>BOUND THEN B:=1; % CHECK ERROR                                      00021600  05:00:002193
217       LPN:=LPN+K;END;                                                                       00021700  05:00:002223
                                                                                              00021300
218       END;                                                                                  00021800  04:00:002236
                                                                                              00019500
219 %     PHASE TWO STARTS HERE                                                                 00021900  03:00:002244
220       IF CN2K>0 THEN BEGIN                                                                  00022000  03:00:002244
221       LP:=KPKM1;                                                                            00022100  04:01:002262
222       LPN:=LP-1;                                                                            00022200  04:01:002266
223       STORE[0 ]:=T[LPN]-GRABONE(DISB[0],0)*(RTR(1,,U[LP])+U[LP]);                           00022300  04:01:002271
224       LOOP CJ:=1,1,CM3 DO BEGIN C:=CJ.[16:42];                                              00022400  04:01:002305
225       LP:=LP+K;LPN:=LPN+K;                                                                  00022500  05:01:002334
226       STORE[CJ]:=T[LPN]-GRABONE(DISB[C],CJ)*(RTR(I,,U[LP])+U[LP])-                          00022600  05:01:002348
227         GRABONE(E[C],CJ)*STORE[CJ-1];                                                       00022700  05:01:002381
228       END;                                                                                  00022800  05:01:002406
                                                                                              00022400
229       LOOP MC:=1,1,CM3 DO BEGIN                                                             00022900  04:01:002414
230       CJ:=CM3-MC;                                                                           00023000  05:01:002438
231       JC:=CJ+1;                                                                             00023100  05:01:002446
232       C:=JC.[16:42];                                                                        00023200  05:01:002453
233       STORE[CJ]:=STORE[CJ]-GRABONE(E[C],JC)*STORE[JC];                                      00023300  05:01:002459
234       END;                                                                                  00023400  05:01:002486
                                                                                              00022900
235       LPN:=K+K-2;                                                                           00023500  04:01:002494
236       LOOP CJ:=0,1,CM3 DO BEGIN % PERFORM OVERRELAXATION                                    00023600  04:01:002502
237       STORE[CJ]:=W*(STORE[CJ]-U[LPN])+U[LPN];                                               00023700  05:01:002526
238       IF OLDMODE AND CN2K GEQ PEN THEN BEGIN                                                00023800  05:01:002546
239       IF ABS(U[LPN]- STORE[CJ])>BOUND THEN B:=1; % CHECK ERROR                              00023900  06:02:002567
240       U[LPN]:=STORE[CJ];END;                                                                00024000  06:02:002601
                                                                                              00023800
241       LPN:=LPN+K;END;                                                                       00024100  05:01:002611
                                                                                              00023600
242       END;                                                                                  00024200  04:01:002624
                                                                                              00022000
243       IF B=0 THEN GO TO DONE;END;                                                           00024300  03:00:002627
                                                                                              00014000
244 DONE: II:=II;                                                                             00024400  02:00:002659
245       CP:=K;                                                                                00024500  02:00:002664
246       LOOP CI:=0,1,CM3 DO LOOP CJ:=0,1,K-1 DO BEGIN                                         00024600  02:00:002668
247       II:=CI.[16:42];                                                                       00024700  03:00:002719
248       U[CP]:= U[CP]*GRABONE(DI[II],CJ);                                                     00024800  03:00:002724
249       T[CP]:=-T[CP]*GRABONE(D[II],CI);                                                      00024900  03:00:002744
250       CP:=CP+1;END;                                                                         00025000  03:00:002765
                                                                                              00024600
251       C:=K*CM3;                                                                             00025100  02:00:002798
252       LOOP CI:=K,1,KPKM1 DO BEGIN                                                           00025200  02:00:002800
253       T[CI]:=T[CI]-ALPH*U[CI-K];                                                            00025300  03:00:002832
254       CP:=CI+C;                                                                             00025400  03:00:002855
255       T[CP]:=T[CP]-ALPH*U[CP+K];END;                                                        00025500  03:00:002863
                                                                                              00025200
256     END.%% OF MSLOR                                                                         00025600  02:00:002897
                                                                                              00006100
```


NUMBER OF ERRORS DETECTED=0000. NUMBER OF WARNINGS NOTED=0000.
SOURCE PROGRAM SIZE= 00256 CARD IMAGES, 003186 SYNTACTIC ITEMS.
SOURCE FILE: CARD=ERICKSEN/PRINT/SLOR.

```
MACRO LIBRARY: ILLIAC/GLYPNIR3/MACROS.
ESTIMATED CORE MEMORY REQUIREMENTS= 001707 WORDS CODE. 000806 WORDS DATA.
COMPILATION TIME= 0034 SECONDS ELAPSED. 0017 SECONDS PROCESSING.
```

PEACEMAN-RACHFORD ADI IN GLYPNIR

I L L I A C I V G L Y P N I R C O M P I L E R

(B6500 VERSION 3.3.00)

```
 1       SUBROUTINE ADI(PCPOINT OUT U,PCPOINT T,CINT CM,CINT CN,CREAL HX,          00000100  00:00:000000
 2       CREAL HY,CINT FLAG,CINT OUT MAXIT,CREAL ERROR);                           00000200  01:00:000012
 3 %                                                                               00000300  01:00:000012
 4 %%%%%%%%%%%%%%%%  DEFINITION OF PARAMETERS  %%%%%%%%%%%%%%%                   00000400  01:00:000012
 5 %                                                                               00000500  01:00:000012
 6 %  U IS THE MESH TO BE IMPROVED                                                 00000600  01:00:000012
 7 %  T IS THE CONSTANT MESH IN POISSON'S EQUATION                                00000700  01:00:000012
 8 %  CM IS THE NUMBER OF COLUMNS IN U                                            00000800  01:00:000012
 9 %  CN IS THE NUMBER OF ROWS IN U                                               00000900  01:00:000012
10 %  HX IS THE DISTANCE BETWEEN ROWS IN U                                        00001000  01:00:000012
11 %  HY IS THE DISTANCE BETWEEN COLUMNS IN U                                     00001100  01:00:000012
12 %  FLAG TELLS THE SUBROUTINE IF THE MAIN PROGRAM SUPPLIED                      00001200  01:00:000012
13 %  THE NUMBER OF ITERATIONS TO BE PERFORMED OR THE                             00001300  01:00:000012
14 %  ERROR BOUND                                                                 00001400  01:00:000012
15 %  MAXIT IS THE NUMBER OF ITERATIONS THAT WILL BE PERFORMED                    00001500  01:00:000012
16 %  IF FLAG=0 THEN MAXIT MUST BE SUPPLIED BY THE MAIN PROGRAM                   00001600  01:00:000012
17 %  ERROR IS THE ERROR BOUND OBTAINED AFTER MAXIT ITERATIONS                    00001700  01:00:000012
18 %  IF FLAG=1 THEN THE MAIN PROGRAM SUPPLIES ERROR AND MAXIT IS                 00001800  01:00:000012
19 %  CALCULATED IN THIS SUBROUTINE                                               00001900  01:00:000012
20 %                                                                              00002000  01:00:000012
21 %%%%%%%   THE USE   %%%%%  SOLVE POISSON'S EQUATION   %%%%%%                  00002100  01:00:000012
22 %                                                                              00002200  01:00:000012
23 %   CONSIDER A RECTANGULAR MESH, U[*], WHICH HAS CN ROWS AND                   00002300  01:00:000012
24 %   CM COLUMNS.  THE VALUES OF THE BOUNDARY ARE GIVEN.  THAT                   00002400  01:00:000012
25 %   IS THE VALUES OF ROW 1, ROW N, COLUMN 1 AND COLUMN M ARE                   00002500  01:00:000012
26 %   KNOWN.  THE CONSTANT MESH T IS ALSO KNOWN.  WE WANT TO                     00002600  01:00:000012
27 %   OBTAIN THE INTERIOR VALUES OF U SUCH THAT:                                 00002700  01:00:000012
28 %   U[I,J]=H*(U[I-1,J]+U[I+1,J])+V*(U[I,J-1]+U[I,J+1])-G*T[I,J]                00002800  01:00:000012
29 %   WHERE A=HY*HY,B=HX*HX,H=A/(2*(A+B)),V=B/(2*(A+B)),AND G=H*B                 00002900  01:00:000012
30 %   THIS PROGRAM IS LIMITED TO MESHES WHERE CN AND CM ARE                      00003000  01:00:000012
31 %   LESS THAN OR EQUAL TO 64.                                                  00003100  01:00:000012
32 %                                                                              00003200  01:00:000012
33 %%%%%   METHOD   %%%%%  ALTERNATING DIRECTION IMPLICITE METHOD %               00003300  01:00:000012
34 %                                                                              00003400  01:00:000012
35 %   WE WILL USE THE PEACEMAN-ROCKFORD SCHEME AS FOUND IN [2] PAGE              00003500  01:00:000012
36 %   174.  WE CALCULATE THE OPTIMAL RELAXATION PARAMETERS FOR ALL THE           00003600  01:00:000012
37 %   ITERATIONS USING METHOD IN [2] PAGE 192.  THEN WE DO THE                   00003700  01:00:000012
38 %   PRECONDITIONING FOR THOMAS' METHOD OF INVERTING TRIDIAGONAL                00003800  01:00:000012
39 %   MATRICES AS FOUND IN [1] PAGE 37.                                          00003900  01:00:000012
40 %   FINALLY WE DO THE ROW RELAXATION FOLLOWED BY THE COLUMN                    00004000  01:00:000012
41 %   RELAXATION.                                                                00004100  01:00:000012
42 %                                                                              00004200  01:00:000012
43 %%%%%%   TYPE OF STORAGE USED FOR THE MESHES   %%%%%%%%%%%%%%%%%%            00004300  01:00:000012
44 %                                                                              00004400  01:00:000012
45 %   U AND T ARE TO BE SUPPLIED TO THE SUBROUTINE IN STRAIGHT                   00004500  01:00:000012
46 %   STORAGE.  BEFORE ANY ITERATIONS ARE PERFORMED U AND T WILL                 00004600  01:00:000012
47 %   WILL BE PLACED IN SCKEWED STORAGE.  AFTER THE LAST ITERATION               00004700  01:00:000012
48 %   U AND T WILL BE RETURNED TO STRAIGHT STORAGE.                              00004800  01:00:000012
49 %                                                                              00004900  01:00:000012
50 %%%%%%   LANGUAGE GLYPNIR   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00005000  01:00:000012
```

```
%                                                                    00005100  01:00:000012
%%%%%%%%%%%%%%%   REFERENCES   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%        00005200  01:00:000012
%                                                                    00005300  01:00:000012
%   [1]  SECTION 2.5 OF THIS PAPER                                   00005400  01:00:000012
%   [2]  WACHPRESS, E. L., "ITERATIVE SOLUTION OF ELLIPTIC           00005500  01:00:000012
%        SYSTEMS AND APPLICATIONS TO NEUTRON DIFFUSION EQUATIONS     00005600  01:00:000012
%        OF REACTOR PHYSICS", RENTICE-HALL,ENGLEWOOD CLIFFS,         00005700  01:00:000012
 %        NEW JERSEY, 1966.                                          00005750  01:00:000012
%   [3]  LAWRIE, D. H., "GLYPNIR PROGRAMMING MANUAL".                00005800  01:00:000012
%        ILLIAC IV DOCUMENT NO. 232. DEPARTMENT OF COMPUTER          00005900  01:00:000012
%        SCIENCE, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN,        00006000  01:00:000012
%        URBANA,ILLINOIS,1970.                                       00006100  01:00:000012
%                                                                    00006200  01:00:000012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%     00006300  01:00:000012
%                                                                    00006400  01:00:000012
        BEGIN                                                        00006500  01:00:000012
        CINT ITE;                                                    00006600  02:00:000012
        BOOLEAN VMODE,HMODE,NEWMODE;                                 00006700  02:00:000012
        PCPOINT B;                                                   00006800  02:00:000012
        PREAL DIA,W,ALBE;                                            00006900  02:00:000012
        PINT D;                                                      00007000  02:00:000012
        CREAL DELTA,AI,BI,CA,ALPH,BETA,AP,BP,KP,M,MAXH,MAXV,MINH,MINV; 00007100  02:00:000012
        CINT IC,CI,CB,IT,IT1,CN2,CM2,CM3,CN3,CK,ITER,ITERA,CIT,CC;   00007200  02:00:000012
        CREAL DIAGONAL,PI,LPI,RABONE;                                00007300  02:00:000012
        PI:=3.1415926536;                                            00010700  02:00:000012
        CN2:=CN-2;                                                   00010800  02:00:000017
        CM2:=CM-2;                                                   00010900  02:00:000024
        CM3:=CM-3;                                                   00011000  02:00:000031
        CN3:=CN-3;                                                   00011100  02:00:000038
        AI:=-1.0*HX*HX;                                              00011200  02:00:000045
        BI:=-1.0*HY*HY;                                              00011300  02:00:000059
        ALPH :=1.0/AI; % CALCULATE HORIZONTAL COEFFICENT             00011400  02:00:000073
        BETA:=1.0/BI;  % CALCULATE VERTICAL  COEFFICENT              00011500  02:00:000084
        MINV:=2.0*SIN(PI/(CN+CN2));                                  00011600  02:00:000095
        MINV:=-1.0*ALPH*MINV*MINV;                                   00011700  02:00:000124
        MAXV:=2.0*SIN(CN2*PI/(4*CN2));                               00011800  02:00:000141
        MAXV:=-1.0*ALPH*MAXV*MAXV;                                   00011900  02:00:000181
        MINH:=SIN(PI/(CM+CM2))*2.0;                                  00012000  02:00:000198
        MINH:=-1.0*BETA*MINH*MINH;                                   00012100  02:00:000225
        MAXH:=SIN(CM2*PI/(4*  CM2))*2.0;                             00012200  02:00:000242
        MAXH:=-1.0*BETA*MAXH*MAXH;                                   00012300  02:00:000280
        M:=2.0*(MAXV-MINV)*(MAXH-MINH)/((MAXH+MAXV)*(MINV+MINH));    00012400  02:00:000297
        KP:=1.0/((1.0+M)+SQRT(M*(2.0+M)));                           00012500  02:00:000340
        IF MAXH=MAXV AND MINH=MINV THEN DELTA:=0.0 ELSE              00012600  02:00:000374
        DELTA:=2.0*(KP*(MAXH+MAXV)-MINH-MINV)/((MINH+MINV)*(MAXH-MAXV)+KP* 00012700  02:01:000419
        (MAXH+MAXV)*(MINV-MINH));                                    00012800  02:01:000456
        AP:=KP*(MINV-MINH+2.0*DELTA*MINH*MINV)/(MINH+MINV);          00012900  02:00:000486
        BP:=(2.0+DELTA*(MAXH-MAXV))/(MAXH+MAXV);                     00013000  02:00:000524
        CA:=1.0/(PI*PI);                                             00013100  02:00:000553
        DIAGONAL:=0.25*KP;                                           00013200  02:00:000571
        DIAGONAL:=LN(DIAGONAL);                                      00013300  02:00:000581
%       IF FLAG EQUALS ONE THEN WE CALULATE THE NUMBER OF ITERATIONS 00013400  02:00:000590
%       REQUIRED TO OBTAIN THE DESIRED ERROR BOUND.  IF FLAG DOESNOT EQUAL 00013500  02:00:000590
%       ONE THEN WE CALCULATE THE ERROR BOUND WE OBTAIN IN MAXIT ITFRATION 00013600  02:00:000590
        IF FLAG = 1 THEN BEGIN % CALCULATE NUMBER OF ITERATIONS      00013700  02:00:000590
        W:=0.25*ERROR;W:=LN(W)*DIAGONAL;                             00013800  03:01:000607
        MAXIT:=GRABONE(W,1)*CA;                                      00013900  03:01:000623
        END ELSE BEGIN % CALCULATE ERROR                             00014000  03:01:000639
                                                            00013700
        W:=MAXIT/(CA*DIAGONAL);W:=EXP(W);                            00014100  03:01:000646
```

```
110       ERROR:=GRABONE(W,1)*4.0;END;                                                    00014200  03:01:000669
                                                                                        00014000
111       ALBE:=ALPH;                                                                     00014300  02:00:000690
112       NEWMODE:=BOOLEAN(5555555555555555(16));                                         00014400  02:00:000694
113       MODE:= NEWMODE;                                                                 00014500  02:00:000699
114       ALBE:=BETA;%ALTERNATING HORIZONTAL AND VERTICAL COEFFICENTS                     00014600  02:00:000703
115       MODE:=TRUE;IF (CM-CN)>0 THEN CB:=CM-4  ELSE CB:=CN-4;                           00014700  02:00:000707
116       B:=GETPEB(CB+2);                                                                00014800  02:00:000756
117       LOOP CI:=1,1,CN-1 DO BEGIN                                                      00014900  02:00:000765
118       U[C1]:=RTR(CI,,U[CI]);                                                          00015000  03:00:000792
119       T[CI]:= RTR(CI,,T[CI]);END;                                                     00015100  03:00:000807
                                                                                        00014900
120       ITER:=MAXIT.[59:5];                                                             00015200  02:00:000830
121       ITE:=ITER+ITER-2;                                                               00015300  02:00:000836
122       ITERA:=MAXIT.[16:43];                                                           00015400  02:00:000846
123 %     THE FOLLOWING IS USED WHEN MAXIT IS A MULTIPLE OF 32                            00015500  02:00:000852
124       IF ITER=0  THEN BEGIN ITE:=62;ITERA:=ITERA-1;END;                               00015600  02:00:000852
                                                                                        00015600
125       IF PEN>0 AND PEN LEQ CN2 THEN  VMODE:=MODE;                                     00015700  02:00:000884
126       IF PEN<CN THEN D:=PEN;                                                          00015800  02:00:000918
127       IF PEN>0 AND PEN LEQ CM2 THEN HMODE:=MODE;                                      00015900  02:00:000941
128       CK:=1;                                                                          00016000  02:00:000975
129       LOOP CIT:=0,1,ITERA DO BEGIN                                                    00016100  02:00:000980
130       W:=(PEN+CK)/(2*MAXIT);                                                          00016200  03:00:001004
131       MODE:=NEWMODE;                                                                  00016300  03:00:001032
132       W:=2.0*(1.0-RTR(1,,W));                                                         00016400  03:00:001036
133       MODE:=TRUE;                                                                     00016500  03:00:001055
134       W:= W*DIAGONAL;                                                                 00016600  03:00:001059
135       W:=EXP(W);                                                                      00016700  03:00:001064
136       MODE:=REVR(1,NEWMODE);                                                          00016800  03:00:001069
137       W:=2.0*W*(1.0+RTL(1,,W))/(1.0+W*W);                                             00016900  03:00:001074
138       MODE:=NEWMODE;                                                                  00017000  03:00:001104
139       W:=RTR(1,,W);                                                                   00017100  03:00:001108
140       W:=(W+AP)/(BP+DELTA*W);                                                         00017200  03:00:001116
141       MODE:=REVR(1,MODE);                                                             00017300  03:00:001133
142       W:=(W-AP)/(BP-DELTA*W);                                                         00017400  03:00:001137
143       MODE:=TRUE;                                                                     00017500  03:00:001155
144       DIA:=W-2.0*ALBE;                                                                00017600  03:00:001159
145       B[0]:=ALBE/DIA;                                                                 00017700  03:00:001167
146       LOOP CI:=0,1,CB  DO                                                             00017800  03:00:001172
147       B[CI+1]:=ALBE/(DIA-ALBE*B[CI]);                                                 00017900  03:00:001196
148       LOOP CC:=0,2,ITE  DO BEGIN                                                      00018000  03:00:001222
149 %     IMPROVE THE ROWS SIMULTANEOUSLY                                                 00018100  04:00:001246
150       IT1:=CC+1;                                                                      00018200  04:00:001246
151       CA:=2.0*ALPH +GRABONE(W,IT1);                                                   00018300  04:00:001252
152       LOOP CI:=0,1,CM3 DO BEGIN                                                       00018400  04:00:001268
153       RABONE:=GRABONE(B[CI],IT1);                                                     00018500  05:00:001292
154       D:=RTR(1,,D);VMODE:=REVR(1,VMODE);MODE:=VMODE;                                  00018600  05:00:001302
155       U[D]:=(RI*(-T[D]-ALPH*(RTL(1,,U[D])+RTR(1,,U[D]))+CA*U[D])-                     00018700  05:00:001319
156       RTR(1,,U[D+1]))* RABONE;                                                        00018800  05:00:001360
157       MODE:=TRUE;END;                                                                 00018900  05:00:001381
                                                                                        00018400
158       LOOP IC:=0,1,CM3 DO BEGIN                                                       00019000  04:00:001393
159       CI:=CM2-IC;RABONE=GRABONE(B[CI-1],IT1);                                         00019100  05:00:001417
160       MODE:=VMODE;                                                                    00019200  05:00:001439
161       U[D]:=U[D]-RTL(1,,U[D-1])*RABONE;                                               00019300  05:00:001443
162       VMODE:=REVL(1,VMODE);MODE:=TRUE;D:=RTL(1,,D);                                   00019400  05:00:001467
163       END;                                                                            00019500  05:00:001484
                                                                                        00019000
164 %     IMPROVE THE COLUMNS SIMULTANEOUSLY                                              00019600  04:00:001492
```

```
165       CA:=2.0*BETA+GRABONE(W,CC);                                                    00019700  04:00:001492
166       LOOP CI:=1,1,CN2 DO BEGIN                                                      00019800  04:00:001507
167       MODE:=TRUE;RABONE:=GRABONE(B[CI-1],CC);                                        00019900  05:00:001531
168       HMODE:=REVR(1,HMODE);                                                          00020000  05:00:001547
169       MODE:=HMODE;                                                                   00020100  05:00:001552
170       U[CI]:=(A1*(-T[CI]-BETA*(RTR(1,,U[CI])+RTL(1,,U[CI])))+CA*U[CI])               00020200  05:00:001556
171       -RTR(1,,U[CI-1]))* RABONE;END;                                                 00020300  05:00:001598
                                                                                 00019800
172       LOOP IC:=0,1,CN3 DO BEGIN                                                      00020400  04:00:001632
173       C1:=CN2-IC;                                                                    00020500  05:00:001656
174       MODE:=TRUE;RABONE:=GRABONE(B[CI-1],CC);                                        00020600  05:00:001664
175       MODE:=HMODE;                                                                   00020700  05:00:001681
176       U[CI]:=U[CI]- RABONE*RTL(1,,U[CI+1]);                                          00020800  05:00:001685
177       HMODE:=REVL(1,HMODE);END;                                                      00020900  05:00:001715
                                                                                 00020400
178       MODE:=TRUE;                                                                    00021000  04:00:001728
179       END;                                                                           00021100  04:00:001732
                                                                                 00018000
180       CK:=CK+ITER;                                                                   00021200  03:00:001740
181       ITE:=62;                                                                       00021300  03:00:001749
182       ITER:=32;END;                                                                  00021400  03:00:001754
                                                                                 00016100
183       LOOP CI:=1,1,CN-1 DO BEGIN                                                     00021500  02:00:001767
184       U[C1]:=RTL(CI,,U[CI]);                                                         00021600  03:00:001794
185       T[CI]:= RTL(CI,,T[CI]);END;                                                    00021700  03:00:001612
                                                                                 00021500
186        END.                                                                          00021800  02:00:001838
                                                                                 00006500
```

```
NUMBER OF ERRORS DETECTED=0000.  NUMBER OF WARNINGS NOTED=0000.
SOURCE PROGRAM SIZE= 00186 CARD IMAGES, 002145 SYNTACTIC ITEMS.
SOURCE FILE: CARD=ERICKSEN/PRINT/ADI.
MACRO LIBRARY: ILLIAC/GLYPN1R3/MACROS.
ESTIMATED CORE MEMORY REQUIREMENTS= 001084 WORDS CODE, 000684 WORDS DATA.
COMPILATION TIME= 0037 SECONDS ELAPSED, 0015 SECONDS PROCESSING.
```

```
 1    BEGIN                                                                    00000100  00:00:000000
 2       PE INTEGER SUBROUTINE TURN AS RGA(CINT P, PINT I);                    00000200  01:00:000000
 3       BEGIN                                                                 00000300  02:00:000000
 4         RGA  := I ; CAR1  := P ;                                            00000400  03:00:000000
 5         CODE BEGIN                                                          00000500  03:00:000003
 6                LIT(2)      =1,0,1 ;                                       00000600  03:00:000003
 7                SHABR       0(1) ;                                         00000700  03:00:000003
 8                CSHL(1)     24;                                            00000800  03:00:000003
 9                CLRA ;                                                     00000900  03:00:000003
10                COR(1)     $C2 ;                                           00001000  03:00:000003
11                SHABL       1 ;                                            00001100  03:00:000003
12                RTAR        2 ;                                            00001200  03:00:000003
13                TXEFAM(1)  ,-3;                                            00001300  03:00:000003
14                CSHR(1)    24 ;                                            00001400  03:00:000003
15                RTAL        1(I);                                          00001500  03:00:000003
16         END CODE ;                                                        00001600  03:00:000003
17       END ; % OF TURN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00001700  03:00:000003
                                                                            00000300
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00001800  01:00:000015
19       SUBROUTINE  SETPTI(PCPOINT A, PCPOINT S, PCPOINT ISS,CINT OUT PP     00001900  01:00:000015
20       ,PCPOINT INDEX,CINT NX,CREAL HX,CREAL HY) ;                          00002000  02:00:000015
21 %     SETPTI CALCULATES FOUR PARAMETERS REQUIRED BY                        00002100  02:00:000015
22 %     POISSOND1RECT, A,S,ISS,INDEX.  THESE PARAMETERS                      00002200  02:00:000015
23 %     ARE VECTORS WHICH REQUIRE CN/128 LINES OF PEM                        00002300  02:00:000015
24 %     EACH.  CN IS THE NUMBER OF ROWS IN THE MESH, NOT                     00002400  02:00:000015
25 %     PEM ROWS. NX EQUALS CN-1.  HX AND HY ARE                             00002500  02:00:000015
26 %     THE HORIZONTAL AND VERTICAL COEFFICIENTS                             00002600  02:00:000015
27 %     RESPECTIVELY.  PP IS THE LOG BASE 2 OF NX.                           00002700  02:00:000015
28 %     SETPTI MUST BE CALLED EVERY TIME THE MESH SIZE CHANGES               00002800  02:00:000015
29 %     OR HX AND(OR) HY CHANGE.                                             00002900  02:00:000015
30    BEGIN                                                                   00003000  02:00:000015
31       CINT I,R,KN ;                                                        00003100  03:00:000015
32       PINT L,Q ;                                                           00003200  03:00:000015
33       CREAL PI,H,X ;                                                       00003300  03:00:000015
34       PI  := 3.1415926535898 ;                                             00003400  03:00:000015
35       PP := LN(NX)/LN(2);                                                  00003500  03:00:000020
36       L  := PEN ;                                                          00003600  03:00:000051
37       H:=LN(2);SIMWRITE(LINE,"LN(2)",H);                                   00003700  03:00:000053
38       X := PI / NX ;                                                       00003800  03:00:000084
39       H  := HX / HY ;                                                      00003900  03:00:000101
40       KN := NX.[16:41] ;                                                   00004000  03:00:000111
41       LOOP 1  := 0,1,KN DO                                                 00004100  03:00:000117
42       BEGIN                                                                00004200  03:00:000141
43         S[1]  := SIN(L*X) ;                                                00004300  04:00:000141
44         ISS[I]  := S[I] *S[I] ;                                            00004400  04:00:000157
45         IF ISS[I] NEQ 0 THEN BEGIN ISS[I]  := -1.0 / ISS[I] ; END;         00004500  04:00:000170
                                                                            00004500
46         A[I]  := -2.0 * (H- H * COS(L * X) + 1 ) ;                         00004600  04:00:000210
47         INDEX[I]  := TURN(PP-1,L);                                         00004700  04:00:000248
48         L  := L + 64 ;                                                     00004800  04:00:000277
```

```
49        END ;

50      END ;

51          SUBROUTINE
52          POISSONDIRECT(PCPOINT P,PCPOINT C,PCPOINT A,PCPOINT S,CINT CN,
53          CINT CM,CREAL HX,CREAL HY,PCPOINT ISS,
54          CINT LLN,PCPOINT INDEX ) ;
55  %%%%%%%%%%%%%%%%%  FUNCTION  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56  %
57  %       THIS SUBROUTINE SOLVES POISSON' EQUATION USING A DIRECT
58  %       METHOD SIMILAR TO HOCKNEY'S PROGRAM [1].
59  %       C[I,J) = HX*(P[I+1,J]+P[I-1,J]-2*P[I,J] +
60  %                HY*(P[I,J+1]+P[I,J-1]-2*P[I,J]).
61  %       THE CHARGE, C, AND THE BOUNDARY POINTS OF THE POTENTIAL,
62  %       P, ARE USED TO CALCULATE THE INTERIOR POINTS OF POTENTIAL.
63  %
64  %%%%%%%%%%%%%%%%  PARAMETERS  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65  %
66  %       P IS THE POTENTIAL.  C IS THE CHARGE.
67  %       A CONTAINS THE DIAGONAL ELEMENTS OF THE TRIDIAGONAL
68  %       MATRICES SOLVED IN THE SECTION OF THIS ROUTINE CALLED
69  %       CRED.  S CONTAINS THE VALUES OF THE SIN WHICH ARE NEEDED
70  %       BY THE FOURIER TRANSFORM PART OF THE ALGORITHM.  ISS IS
71  %       ALSO NEEDED BY FOURIER.
72  %       INDEX IS USED FOR SCRAMBLING BY FOURIER ANALYSIS AND
73  %       SYNTHESIS.  A, S, ISS, AND INDEX ARE CALCULATED BY SETPT1.
74  %       P HAS CN ROWS AND CM COLUMNS.  HX(HY) IS THE HORIZONTAL(
75  %       VERTICAL) COEFFICENT OF POISSON'S EQUATION.  LLN EQUALS
76  %       THE LOG BASE 2 OF CN-1.
77  %
78  %%%%%%%%%%%%%%%%  METHOD  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
79  %
80  %       A FOURIER ANALYSIS IS DONE TO EACH COLUMN BY SUBROUTINE
81  %       FOURIER.  THIS IS FOLLOWED BY SOLVING THE TRIDIAGONAL
82  %       MATRICES, ONE PER ROW.  FINALLY A FOURIER SYNTHESIS IS
83  %       PERFORMED ON EACH ROW TO GIVE ANSWER.  FOR FURTHER DETIAL
84  %       SEE SECTIONS 3, 4 AND 5 OF THIS PAPER.
85  %
86  %%%%%%%%%%%%%%%%%%%%  REFERENCES  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
87  %
88  %       [1]  R.W. HOCKNEY, "THE POTENTIAL CALCULATION AND
89  %            SOME APPLICATIONS", METHODS IN COMPUTATIONAL
90  %            PHYSICS, VOL. 9, P. 136-211, 1970.
91  %       [2]  J. W. COOLEY, ET AL, "THE FAST FOURIER TRANSFORM
92  %            ALGORITHM:  PROGRAMMING CONSIDERATIONS IN THE
93  %            CALCULATION OF SINE, COSINE AND LAPLACE TRANSFORMS"
94  %            JOURNAL OF SOUND AND VIBRATION, VOLUMN 12 NUMBER 2,
95  %            JUNE 1970.
96      BEGIN
97          CINT CM2,CN2,K,CNIK,CJ,CL,IL,L1,L2,L3,L4,KM1,KP,KC,I,
98          L,OFF,GL,II,J,N2,ILJ,KM,N3,IT,IP;
99          LABEL FOURIER,FINISH ;
100         PINT PI,PJ;
101         CREAL POTFAC,RR,RR1,CO,SI; BOOLEAN FMODE,SMODE,SAVEMODE ;
102         PCPOINT STORE ;
103       PREAL PR,0001,00D2,0DD3 ;
104         PREAL E,Q,T;
105 %%%%%%%%%%%%%%%%  SECTION 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
106       SAVEMODE:=MODE; MODE:=TRUE;
```

```
00004900  04:00:000280
00004200
00005000  03:00:000288
00003000
00005100  01:00:000306
00005200  01:00:000306
00005300  02:00:000306
00005400  02:00:000306
00005500  02:00:000306
00005600  02:00:000306
00005700  02:00:000306
00005800  02:00:000306
00005900  02:00:000306
00006000  02:00:000306
00006100  02:00:000306
00006200  02:00:000306
00006300  02:00:000306
00006400  02:00:000306
00006500  02:00:000306
00006600  02:00:000306
00006700  02:00:000306
00006800  02:00:000306
00006900  02:00:000306
00007000  02:00:000306
00007100  02:00:000306
00007200  02:00:000306
00007300  02:00:000306
00007400  02:00:000306
00007500  02:00:000306
00007600  02:00:000306
00007700  02:00:000306
00007800  02:00:000306
00007900  02:00:000306
00008000  02:00:000306
00008100  02:00:000306
00008200  02:00:000306
00008300  02:00:000306
00008400  02:00:000306
00008500  02:00:000306
00008600  02:00:000306
00008700  02:00:000306
00008800  02:00:000306
00008900  02:00:000306
00009000  02:00:000306
00009100  02:00:000306
00009200  02:00:000306
00009300  02:00:000306
00009400  02:00:000306
00009500  02:00:000306
00009600  02:00:000306
00009700  03:00:000306
00009800  03:00:000306
00009900  03:00:000306
00010000  03:00:000306
00010100  03:00:000306
00010200  03:00:000306
00010300  03:00:000306
00010400  03:00:000306
00010500  03:00:000306
00010600  03:00:000306
```

```
107         K := CM .[16:42] ;                                                    00010700  03:00:000312
108         I := 0 ;                                                              00010800  03:00:000318
109         IF CM.[58:6] GTR 0 THEN 1 := 1 ;                                      00010900  03:00:000323
110         CN2 := CN - 2 ; POTFAC := 0.125 / ((CN-1)*HY) ;                       00011000  03:00:000349
111         CM2 := CM - 2 ;                                                       00011100  03:00:000380
112         K := K + 1 ;                                                          00011200  03:00:000387
113         KM1 := K - 1 ;                                                        00011300  03:00:000396
114         KP := K + K ;                                                         00011400  03:00:000403
115         STORE := GETPEB(KP+K) ;                                               00011500  03:00:000412
116         KC := K * CN2 ; PI := 0 ;                                             00011600  03:00:000423
117         1 := CM.[58:6] ;                                                      00011700  03:00:000437
118         OFF := 1 - 2 ;                                                        00011800  03:00:000443
119         IF 1 = 0 THEN BEGIN OFF := 62 ; END;                                  00011900  03:00:000450
                                                                                00011900
120       MODE :=  ( PEN GTR OFF ) ;                                              00012000  03:00:000473
121         P1 := -1 ;                                                          00012100  03:00:000486
122         MODE := TRUE ;                                                      00012200  03:00:000492
123         CJ := KC + 1 ;                                                      00012300  03:00:000496
124         LOOP I := 0,1,KM1 DO                                                00012400  03:00:000503
125         BEGIN                                                               00012500  03:00:000527
126           STORE[I] := C[I+K] ;                                              00012600  04:00:000527
127           STORE[I+K] := C[1+KC] ;                                           00012700  04:00:000540
128           STORE[I+KP] := P[I] ;                                             00012800  04:00:000559
129         END ;                                                               00012900  04:00:000573
                                                                                00012500
130         LOOP 1  := 1,1,KM1 DO                                               00013000  03:00:000581
131         BEGIN                                                               00013100  03:00:000605
132           C[P1+CJ] := C[P1+CJ] - P[CJ+PI+K] * HX ;                          00013200  04:00:000605
133           CJ := CJ +1 ;                                                     00013300  04:00:000639
134           C[PI+K+I] := C[PI+K+I] - P[PI+I] * HX ;                           00013400  04:00:000646
135         END;                                                                00013500  04:00:000680
                                                                                00013100
136       IF PEN GTR 0 AND PEN LSS OFF+1 THEN                                   00013600  03:00:000688
137       BEGIN                                                                 00013700  03:00:000711
138         C[K] := C[K] -P[0] * HX ;                                           00013800  04:01:000717
139         C[KC] := C[KC] - P[KC+K] * HX ;                                     00013900  04:01:000734
140         FMODE := MODE ;                                                     00014000  04:01:000760
141       END ;                                                                 00014100  04:01:000762
                                                                                00013700
142       PJ := 0 ;                                                             00014200  03:00:000764
143       PI := 0 ;                                                             00014300  03:00:000766
144       MODE :=  ( PEN EQL OFF  ) ;                                           00014400  03:00:000768
145       P1 := KM1 ;                                                           00014500  03:00:000780
146          MODE := TRUE ;                                                     00014600  03:00:000784
147       MODE := ( PEN GTR OFF-1 ) ;                                           00014700  03:00:000788
148       PJ := -1 ;                                                            00014800  03:00:000802
149       MODE := TRUE ;                                                        00014900  03:00:000808
150       IF OFF = 1 THEN BEGIN FMODE :=(PEN GTR 0); END;                       00015000  03:00:000812
                                                                                00015000
151       LOOP 1 := K,K,KC DO                                                   00015100  03:00:000841
152       BEGIN                                                                 00015200  03:00:000882
153         PR := 0.0 ;                                                         00015300  04:00:000882
154       MODE :=  ( PEN EQL OFF ) ;                                            00015400  04:00:000887
155         PR := RTL(1,,P[I+KM1]);                                             00015500  04:00:000899
156          MODE := TRUE ;                                                     00015600  04:00:000915
157       MODE := PEN EQL 1 ;                                                   00015700  04:00:000919
158         PR := RTR(1,,P[I]) + PR ;                                           00015800  04:00:000929
159          MODE := TRUE ;                                                     00015900  04:00:000941
160         MODE := FMODE ;                                                     00016000  04:00:000945
161         P[1+PI] :=(C[1+P1] -PR*HY) * POTFAC ;                               00016100  04:00:000949
```

```
162         MODE := TRUE ;                                                  00016200  04:00:000971
163         LOOP L1:= 1,1,KM1 DO                                            00016300  04:00:000975
164         BEGIN                                                           00016400  04:00:000999
165           P[L1+I+PJ] :=  C[L1+I+PJ] * POTFAC ;                          00016500  05:00:000999
166         END;                                                            00016600  05:00:001020
                                                                          00016400
167       END ;                                                             00016700  04:00:001028
                                                                          00015200
168       J  :=(CN-1)*K;                                                    00016800  03:00:001036
169       L:=J.[16:47];                                                     00016900  03:00:001049
170       II:= 0;                                                           00017000  03:00:001055
171 %%%%%%%%%%%%%  SECTION 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00017100  03:00:001060
172 %     FOURIER ANALYSIS AND SYNTHESIS                                    00017200  03:00:001060
173  FOURIER: II := II +1 ;                                                 00017300  03:00:001060
174       PI  :=0;                                                          00017400  03:00:001068
175           IF  ( PEN EQL OFF+1 ) THEN BEGIN                              00017500  03:00:001070
176       PI  := -1 ;    END ;                                              00017600  04:01:001087
                                                                          00017500
177       MODE  := FMODE ;                                                  00017700  03:00:001095
178       KM1  := K-1 ;                                                     00017800  03:00:001099
179       KP  := K + K ;                                                    00017900  03:00:001106
180       N3  := CN - 1 ;                                                   00018000  03:00:001115
181       N2 := N3.[16:47];                                                 00018100  03:00:001122
182       LOOP ILJ  := 0,1,KM1 DO                                           00018200  03:00:001128
183       BEGIN                                                             00018300  03:00:001152
184         KM  := K + ILJ ;                                                00018400  04:00:001152
185         LOOP I  := K,K,L-K DO % LOOP NUMBER 1                           00018500  04:00:001160
186         BEGIN                                                           00018600  04:00:001206
187           IL  := I - ILJ ;                                              00018700  05:00:001206
188           ODD1  := P[I+ILJ+PI];                                         00018800  05:00:001213
189           P[PI+I+ILJ]  := ODD1 + P[PI+J-IL] ;                           00018900  05:00:001223
190           P[PI+J-IL]  := ODD1 - P[PI+J-IL] ;                            00019000  05:00:001244
191         END; % OF LOOP NUMBER 1                                         00019100  05:00:001267
                                                                          00018600
192             P[PI+L+]LJ]  := P[PI+L+ILJ] + P[PI+L+ILJ] ;                 00019200  04:00:001275
193         IL  := K - ILJ ;                                                00019300  04:00:001308
194         P[PI+1LJ]  := - P[PI+J-IL] - P[PI+J-IL] ;                       00019400  04:00:001316
195         ODD3 := P[PI+KM];                                               00019500  04:00:001349
196         P[PI+KM] := -ODD3 - ODD3 ;                                      00019600  04:00:001358
197         IL := 0 ;                                                       00019700  04:00:001369
198         LOOP I := KP,KP,L-KP DO  % LOOP NUMBER 2                        00019800  04:00:001374
199         BEGIN                                                           00019900  04:00:001420
200           IL := IL + 2 ;                                                00020000  05:00:001420
201           CJ := I + ILJ ;                                               00020100  05:00:001427
202           ODD1 := P[PI+I+KM] - ODD3 ;                                   00020200  05:00:001434
203           GL := IL .[16:42] ;                                           00020300  05:00:001446
204           SMODE := MODE ; MODE := TRUE ;                                00020400  05:00:001452
205           RR := GRABONE(S[GL],IL) ;                                     00020500  05:00:001458
206           GL := (N2-IL).[16:42];                                        00020600  05:00:001469
207           RR1 := GRABONE(S[GL],N2-IL);                                  00020700  05:00:001480
208           MODE := SMODE ;                                               00020800  05:00:001496
209           ODD2 := RR * P[PI+CJ] - RR1 * ODD1 ;                          00020900  05:00:001500
210           ODD1 := RR1 * P[PI+CJ] + RR * ODD1 ;                          00021000  05:00:001519
211           P[PI+CJ] := P[PI+J+KM-I] - P[PI+J-I-K+ILJ] ;                  00021100  05:00:001537
212           ODD3 := P[PI+I+KM] ;                                          00021200  05:00:001576
213           P[PI+I+KM] := ODD2 - P[PI+J-I+ILJ];                           00021300  05:00:001587
214           P[PI+J-I+KM] := ODD2 + P[PI+J-I+ILJ];                         00021400  05:00:001610
215           P[PI+J-I+ILJ] := P[PI+CJ] - ODD1 ;                            00021500  05:00:001636
216           P[PI+CJ] := P[PI+CJ] + ODD1 ;                                 00021600  05:00:001657
217         END ; % OF LOOP NUMBER 2                                        00021700  05:00:001674
```

```
218         ODD1 := P[PI+L+KM] ;                                                00019900
219         P[PI+L+KM] := P[PI+L+1LJ] ;                                                 00021800  04:00:001682
220         P[PI+L+1LJ] := ODD1 + ODD1 ;                                                00021900  04:00:001694
221         LOOP I := 1LJ,K,L-K+1LJ DO  % LOOP NUMBER 3                                 00022000  04:00:001716
222         BEGIN                                                                       00022100  04:00:001728
223           ODD1 := P[PI+I] ;                                                         00022200  04:00:001775
224           P[PI+I] := ODD1 + P[PI+L+I] ;                                             00022300  05:00:001775
225           P[PI+L+I] := ODD1 - P[PI+L+1] ;                                           00022400  05:00:001783
226         END; % OF LOOP NUMBER 3                                                     00022500  05:00:001801
                                                                                        00022600  05:00:001822
227         MODE := TRUE ;                                                      00022200
228       END;                                                                          00022700  04:00:001830
                                                                                        00022800  04:00:001834
229       % SCRAMBLING                                                        00018300
230       IL := 0 ;                                                                     00022900  03:00:001842
231       KM :=K+KM1      ;                                                           00023000  03:00:001842
232       LOOP I := KP,KP,J-KP DO                                                       00023100  03:00:001847
233       BEGIN                                                                       00023200  03:00:001856
234         IL := IL + 1 ;                                                            00023300  03:00:001902
235         GL := IL .[16:42] ;                                                       00023400  04:00:001902
236       ODD3 := INDEX[GL] ;                                                         00023500  04:00:001909
237        CJ :=  GRABONE(ODD3      ,IL) ;                                          00023600  04:00:001915
238         IF IL LSS CJ THEN                                                       00023700  04:00:001921
239         BEGIN                                                                   00023800  04:00:001933
240           CJ := CJ * KP ;                                                       00023900  04:00:001946
241           LOOP 1LJ := 0,1,KM DO  % LOOP NUMBER 4                                00024000  05:01:001952
242           BEGIN                                                                 00024100  05:01:001964
243             ODD1 := P[   1+1LJ] ;                                               00024200  05:01:001988
244             P[I+1LJ] := P [CJ+1LJ] ;                                            00024300  06:01:001988
245             P[CJ+1LJ] := ODD1   ;                                               00024400  06:01:001997
246           END ; % OF LOOP NUMBER 4                                              00024500  06:01:002015
                                                                                        00024600  06:01:002024
247         END ;                                                               00024200
                                                                                        00024700  05:01:002032
248       END ;                                                                 00023900
                                                                                        00024800  04:00:002034
249       IT := KP + KP ;                                                     00023300
250       IP := N3 ;                                                                    00024900  03:00:002042
251       LOOP 1 := 2,1,LLN  -1 DO                                                    00025000  03:00:002051
252       BEGIN                                                                       00025100  03:00:002055
253         IP:=IP.[16:47] ;                                                          00025200  03:00:002082
254         CL := 0 ;                                                                 00025300  04:00:002082
255         L2 := 1 ;                                                                 00025400  04:00:002088
256         CJ := IT.[16:47] ;                                                        00025500  04:00:002093
257         IT := IT + IT ;                                                           00025600  04:00:002098
258         LOOP IL := 0,CJ,CJ DO                                                     00025700  04:00:002104
259         BEGIN                                                                     00025800  04:00:002113
260           LOOP KM := IL,KP,IL+CJ-KP DO                                            00025900  04:00:002146
261           BEGIN                                                                   00026000  05:00:002146
262             GL := (N2-CL) .[16:42] ;                                              00026100  05:00:002193
263             CO := GRABONE(S[GL],N2-CL) * L2 ;                                     00026200  06:00:002193
264             GL := CL .[16:42] ;                                                   00026300  06:00:002204
265             S1 := GRABONE(S[GL],CL) ;                                             00026400  06:00:002232
266             LOOP L3 := 0,1T,J-1T DO                                               00026500  06:00:002238
267             BEGIN                                                                 00026600  06:00:002249
268               MODE := FMODE ;                                                     00026700  06:00:002287
269               LOOP 1LJ := 0,1,KM1 DO % LOOP NUMBER 5                              00026800  07:00:002287
270               BEGIN                                                               00026900  07:00:002291
271                 L1 := L3 + KM + 1LJ ;                                           00027000  07:00:002315
                                                                                        00027100  08:00:002315
```

```
272                 L4 := L1 + CJ + CJ ;                                                    00027200  08:00:002324
273                 ODD1 := P[PI+L4] * C0 - P[PI+L4+K] * SI ;                               00027300  08:00:002336
274                 ODD2 := P[PI+L4+K] * C0 + P[PI+L4] * SI ;                               00027400  08:00:002365
275                 P[PI+L4] := P[PI+L1] - ODD1 ;                                           00027500  08:00:002393
276                 P[PI+L4+K] := P[PI+L1+K] - ODD2 ;                                       00027600  08:00:002410
277                 P[PI+L1] := P[PI+L1] + ODD1 ;                                           00027700  08:00:002433
278                 P[PI+L1+K] := P[PI+L1+K] + ODD2 ;                                       00027800  08:00:002450
279                 MODE := TRUE ;                                                          00027900  08:00:002473
280               END ; % OF LOOP NUMBER 5                                                  00028000  08:00:002477
                                                                                          00027000
281             END ;                                                                       00028100  07:00:002485
                                                                                          00026700
282             CL := CL + IP ;                                                           00028200  06:00:002493
283           END ;                                                                       00028300  06:00:002502
                                                                                          00026100
284           L2 := -1 ;                                                                00028400  05:00:002510
285           IP := - IP ;                                                              00028500  05:00:002516
286         END ;                                                                       00028600  05:00:002522
                                                                                          00025900
287       END;                                                                          00028700  04:00:002530
                                                                                          00025200
288       I := 0 ;                                                                      00028800  03:00:002538
289       LOOP L1 := 1,1,N2 DO                                                          00028900  03:00:002543
290       BEGIN                                                                         00029000  03:00:002567
291         I := I + K ;                                                                00029100  04:00:002567
292         GL := L1 .[16:42] ;                                                         00029200  04:00:002576
293         RR := GRABONE(ISS[GL],L1);                                                  00029300  04:00:002581
294         MODE := FMODE ;                                                             00029400  04:00:002591
295         LOOP ILJ := 0,1,KM1 DO  % LOOP NUMBER 6                                     00029500  04:00:002595
296         BEGIN                                                                       00029600  04:00:002619
297           ODD1 := P[PI+I+ILJ] - P[PI+J-I+ILJ] ;                                     00029700  05:00:002619
298           ODD2 := ( P[PI+I+ILJ] + P[PI+J-I+ILJ] ) * RR ;                            00029800  05:00:002645
299           P[PI+I+ILJ] := ODD2 + ODD1 ;                                              00029900  05:00:002674
300           P[PI+J-I+ILJ] := ODD2 - ODD1 ;                                            00030000  05:00:002686
301           MODE := TRUE ;                                                            00030100  05:00:002701
302         END ; % OF LOOP NUMBER 6                                                    00030200  05:00:002705
                                                                                          00029600
303       END ;                                                                         00030300  04:00:002713
                                                                                          00029000
304 %     END OF FOURIER ANALYSIS AND SYNTHESIS                                         00030400  03:00:002721
305       IF II=2 THEN GO TO FINISH ;                                                   00030500  03:00:002721
306 %%%%%%%%%%%%%%  SECTION 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%                  00030600  03:00:002742
307   %   SKEW POTENTIAL                                                              00030700  03:00:002742
308       I := 0 ;                                                                    00030800  03:00:002742
309       LOOP CJ := KP,K,KC DO % LOOP NUMBER 7                                       00030900  03:00:002747
310       BEGIN                                                                       00031000  03:00:002788
311         I := I + 1 ;                                                              00031100  04:00:002788
312         LOOP IL := CJ,1,CJ+KM1 DO                                                 00031200  04:00:002795
313         BEGIN                                                                     00031300  04:00:002830
314           P[IL] := RTR(I,,P[IL]) ;                                                00031400  05:00:002830
315         END ;                                                                     00031500  05:00:002846
                                                                                          00031300
316       END; % OF LOOP NUMBER 7                                                       00031600  04:00:002854
                                                                                          00031000
317 %     CRED STARTS HERE                                                            00031700  03:00:002862
318       MODE := (PEN LSS CN-1) ;                                                    00031800  03:00:002862
319       PI := PEN * K ;                                                             00031900  03:00:002876
320   %   SINCE P[0] TO P[KM1] IS ALREADY GARBAGE WE WONT PROTECT IT                  00032000  03:00:002884
321       CNIK := CN.[16:42] - 1 ;IF CNIK LSS 0 THEN CNIK:=0;                         00032100  03:00:002884
322       CJ := CM.[58:6] ;                                                           00032200  03:00:002917
```

```
323       IF CJ=0 THEN CJ := 64 ;                                          00032300  03:00:002923
324       LOOP I := 0,1,CNIK DO                                            00032400  03:00:002945
325       BEGIN                                                            00032500  03:00:002970
326         IL := 4 ;                                                      00032600  04:00:002970
327         L4 := 0 ;                                                      00032700  04:00:002975
328         E := 1.0 / A[I] ;                                              00032800  04:00:002980
329         Q := P[PI] * E ;                                               00032900  04:00:002990
330         CL := 64 ;                                                     00033000  04:00:002995
331         LOOP L1 := 0,1,KM1 DO                                          00033100  04:00:003000
332         BEGIN                                                          00033200  04:00:003024
333           IF L1=KM1 THEN CL :=  CJ ;                                   00033300  05:00:003024
334           LOOP L2 := IL,1,CL DO % LOOP NUMBER 8                        00033400  05:00:003047
335           BEGIN                                                        00033500  05:00:003079
336             MODE := REVR(1,MODE) ;                                     00033600  06:00:003079
337             A[I] := RTR(1,,A[I]) ;                                     00033700  06:00:003083
338             E := 1.0 / ( A[I] - RTR(1,,E) ) ;                          00033800  06:00:003097
339             PI := RTR(1,,PI) ;                                         00033900  06:00:003119
340             Q := (P[PI+L1+L4] - RTR(1,,Q)) * E ;                       00034000  06:00:003127
341             L4 := 0 ;                                                  00034100  06:00:003149
342           END :% OF LOOP NUMBER 8                                      00034200  06:00:003154
                                                                    00033500
343           L4 := -1 ;                                                   00034300  05:00:003162
344           IL := 1 ;                                                    00034400  05:00:003168
345         END;                                                           00034500  05:00:003173
                                                                    00033200
346         CL := 64 ;                                                     00034600  04:00:003181
347         E := P[PI+KM1] - A[I] * Q ;                                    00034700  04:00:003186
348         P[PI+KM1] := Q ;                                               00034800  04:00:003202
349         IL := 4 ;                                                      00034900  04:00:003210
350         L4 := 0 ;                                                      00035000  04:00:003215
351         LOOP L1 := 0,1,KM1 DO                                          00035100  04:00:003220
352         BEGIN                                                          00035200  04:00:003244
353           IF L1=KM1 THEN CL := CJ ;                                    00035300  05:00:003244
354           L3 := KM1 - L1 ;                                             00035400  05:00:003267
355           LOOP L2 := IL,1,CL DO % LOOP NUMBER 9                        00035500  05:00:003275
356           BEGIN                                                        00035600  05:00:003307
357             MODE := REVL(1,MODE) ;                                     00035700  06:00:003307
358             PI := RTL(1,,PI) ;                                         00035800  06:00:003311
359             A[I] := RTL(1,,A[I]) ;                                     00035900  06:00:003319
360             Q := RTL(1,,Q) ;                                           00036000  06:00:003333
361             E := RTL(1,,E) ;                                           00036100  06:00:003341
362             T := P[PI+L3+L4] - Q - A[I] * E ;                          00036200  06:00:003349
363             Q := E ;                                                   00036300  06:00:003370
364             E := T ;                                                   00036400  06:00:003372
365             P[PI+L3+L4] := Q ;                                         00036500  06:00:003374
366             L4 := 0 ;                                                  00036600  06:00:003385
367           END : % OF LOOP NUMBER 9                                     00036700  06:00:003390
                                                                    00035600
368           IL := 1 ; L4 := 1 ;                                          00036800  05:00:003398
369         END ;                                                          00036900  05:00:003408
                                                                    00035200
370         PI := PI + 64 * K ;                                            00037000  04:00:003416
371       END ;                                                            00037100  04:00:003425
                                                                    00032500
372       MODE := TRUE ;%  UNSKEW POTENTIAL                                00037200  03:00:003433
373       I := 0 ;                                                         00037300  03:00:003437
374       LOOP CJ := KP,K,KC  DO                                           00037400  03:00:003442
375       BEGIN                                                            00037500  03:00:003483
376         I := I + 1 ;                                                   00037600  04:00:003483
377         LOOP IL := CJ,1,CJ+KM1 DO %LOOP NUMBER 10                      00037700  04:00:003490
```

```
378         BEGIN                                                                  00037800  04:00:003525
379           P[IL] := RTL(I,,P[IL]) ;                                             00037900  05:00:003525
380         END: % OF LOOP NUMBER 10                                               00036000  05:00:003544
                                                                                 00037800
381       END;                                                                     00038100  04:00:003552
                                                                                 00037500
382       GO TO FOURIER ;                                                          00038200  03:00:003560
383 %%%%%%%%%%%%%  SECTION 4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%                       00038300  03:00:003561
384   FINISH:    LOOP I := 0,1,KM1 DO                                            00038400  03:00:003561
385       BEGIN                                                                  00038500  03:00:003586
386         C[1+K] := STORE[I] ;                                                 00038600  04:00:003586
387         C[I+KC] := STORE[I+K] ;                                              00038700  04:00:003600
388         P[I] := STORE[I+KP] ;                                                00038800  04:00:003619
389       END ;                                                                  00038900  04:00:003632
                                                                                 00038500
390       MODE := SAVEMODE ;                                                       00039000  03:00:003640
391     END ;                                                                      00039100  03:00:003644
                                                                                 00009600
392 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00039200  01:00:003662
393 %%%%%%%%%%%%%%%%%%%% A TEST PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00039300  01:00:003662
394       CINT CN,CM,PP,I,J,CK,CS,OFF ;                                            00039400  01:00:003662
395       CREAL HX,HY;                                                             00039500  01:00:003662
396       PREAL  VECTOR INDEX[1];                                                  00039600  01:00:003662
397       PREAL VECTOR AX,SX,ISX[1]; PREAL VECTOR P, C[30];                        00039700  01:00:003662
398       PREAL X ;                                                                00039800  01:00:003662
399       SUBROUTINE DATA;                                                         00039900  01:00:003662
400 %     THIS IS A DUMMY SUBROUTINE USED TO INITIALIZE P,                         00040000  02:00:003662
401 %     THE POTENTIAL.                                                           00040100  02:00:003662
402       CODE                                                                     00040200  02:00:003662
403       USE P ;                                                                  00040300  02:00:003662
404       BEGIN                                                                    00040400  02:00:003663
405       HERE:                                                                    00040500  02:00:003663
406         SET;                                                                   00040600  02:00:003663
407         ORG P;                                                                 00040700  02:00:003663
408         DATA                                                                   00040800  02:00:003663
409 0.50337,0.93489,0.60033,0.44270,0.10798,0.92968,0.92879,0.81380,(0.0)56,       00040900  02:00:003663
410 0.98078,0.52103,0.21337,0.52409,0.60700,0.30343,0.14217,0.33953,(0.0)56,       00041000  02:00:003663
411 0.79932,0.11286,0.55894,0.10389,0.40150,0.79314,0.31545,0.69259,(0.0)56,       00041100  02:00:003663
412 0.34400,0.37663,0.58489,0.45073,0.14357,0.52035,0.12551,0.19098,(0.0)56,       00041200  02:00:003663
413 0.63619,0.06807,0.78106,0.75710,0.66561,0.86359,0.40329,0.02445,(0.0)56,       00041300  02:00:003663
414 0.15432,0.90765,0.08668,0.18121,0.37631,0.54454,0.24602,0.73611,(0.0)56,       00041400  02:00:003663
415 0.95768,0.54112,0.80428,0.75255,0.07879,0.16337,0.07416,0.56654,(0.0)56,       00041500  02:00:003663
416 0.08343,0.76303,0.43107,0.60856,0.50404,0.02133,0.75174,0.97407,(0.0)56;       00041600  02:00:003663
417         ORG HERE;                                                              00041700  02:00:003663
418         END CODE;                                                              00041800  02:00:003663
419 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  00041900  01:00:003675
420       CN := 9;                                                                 00042000  01:00:003675
421       CM := 8 ;                                                                00042100  01:00:003680
422       PP:= 3 ;                                                                 00042200  01:00:003685
423       CK := CN - 1 ;                                                           00042300  01:00:003690
424       HX :=  CK * CK ;                                                         00042400  01:00:003697
425       HY :=(CM-1)*(CM-1) ;                                                     00042500  01:00:003714
426       CS := CK ;                                                               00042600  01:00:003735
427       SETPT1(AX,SX,ISX,PP,INDEX,CK,HX,HY);                                     00042700  01:00:003739
428       LOOP I := 1,1,CS-1 DO BEGIN                                              00042800  01:00:003770
429         C[I] := HX*(P[I-1]+P[I+1]-2*P[I])+HY*(RTL(1,,P[I])                     00042900  02:00:003797
430             +RTR(1,,P[I])-2*P[I]) ;                                            00043000  02:00:003840
431       END ;                                                                    00043100  02:00:003873
                                                                                 00042800
432       LOOP I:=0,1,CS DO SIMWRITE(LINE,"BP[Y]",P[I]) ;                          00043200  01:00:003881
```

```
433       POISSONDIRECT(P,C,AX,SX,CN,CM,HX,HY     ,ISX,PP,INDEX);                00043300  01:00:003933
434       LOOP I := 0,1,CS DO BEGIN                                               00043400  01:00:003968
435         SIMWRITE(LINE,"XP[I]",I,P[I]);                                        00043500  02:00:003992
436       END ;                                                                   00043600  02:00:004022
                                                                          00043400
437     END.                                                                      00043700  01:00:004030
                                                                          00000100
```

NUMBER OF ERRORS DETECTED=0000. NUMBER OF WARNINGS NOTED=0000.
SOURCE PROGRAM SIZE= 00437 CARD IMAGES, 004209 SYNTACTIC ITEMS.
SOURCE FILE: CARD=ERICKSEN/PRINT/HOCKNEYG.
MACRO LIBRARY: ILLIAC/GLYPNIR3/MACROS.
ESTIMATED CORE MEMORY REQUIREMENTS= 002778 WORDS CODE, 009408 WORDS DATA.
COMPILATION TIME= 0141 SECONDS ELAPSED, 0030 SECONDS PROCESSING.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801	UNCLASSIFIED
	2b. GROUP

REPORT TITLE

Iterative and Direct Methods for Solving Poisson's Equation and Their Adaptability to ILLIAC IV

DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Report

AUTHOR(S) (First name, middle initial, last name)

James H. Ericksen

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 20, 1972	114	17

a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DAHC04 72-C-0001	CAC Document No. 60
b. PROJECT NO.	
ARPA Order No. 1899	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

0. DISTRIBUTION STATEMENT

Copies may be requested from the address given in (1) above.

1. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	U. S. Army Research Office - Durham Duke Station, Durham, North Carolina

3. ABSTRACT

DD FORM 1 NOV 68 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Glypnir						
Poisson equation solvers						
ASK						
Numerical integration						
Ordering and partial differential equations						
Computer efficiency and benchmarks						
Input/Output						

LIOGRAPHIC DATA ET	1. Report No. UIUCDCS-R-72-574	2.	3. Recipient's Accession No.
tle and Subtitle: Iterative and Direct Methods for Solving Poisson's Equation and Their Adaptability to ILLIAC IV			5. Report Date: December 20, 1972 6.
uthor(s): James H. Ericksen			8. Performing Organization Rept. No.
erforming Organization Name and Address: Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. DAHC04 72-C-0001
ponsoring Organization Name and Address: U. S. Army Research Office - Durham, Duke Station, Durham, North Carolina			13. Type of Report & Period Covered: Research Report 14.

upplementary Notes

bstracts

This paper examines iterative and direct methods for solving Poisson's equation with regard to their adaptation to ILLIAC IV. SOR, SLOR, ADI, and FACR are programmed in GLYPNIR. Detailed suggestions on ASK code for these methods are also supplied.

FACR, Fourier Analysis and Cyclic Reduction, is the fastest method on rectangular meshes. SOR, Successive Over Relaxation, seems to be the most promising for nonrectangular meshes. The methods are between thirty and forty-five times faster on ILLIAC IV than on a serial machine with speed equal to one of the ILLIAC IV PEs (Processing Elements).

ey Words and Document Analysis. 17a. Descriptors

Glypnir
Poisson equation solvers
ASK
Numerical integration
Ordering and partial differential equations
Computer efficiency and benchmarks
Input/Output

Identifiers/Open-Ended Terms

COSATI Field/Group

ailability Statement: Distribution unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 109
	20. Security Class (This Page) UNCLASSIFIED	22. Price